THE CAMBRIDGE MULTIPLE-ACCESS SYSTEM

USER'S REFERENCE MANUAL

Edited by

D. F. Hartley

UNIVERSITY MATHEMATICAL LABORATORY
CAMBRIDGE

THE CAMBRIDGE MULTIPLE-ACCESS SYSTEM

USER'S REFERENCE MANUAL

Edited by

D. F. Hartley

UNIVERSITY MATHEMATICAL LABORATORY
CAMBRIDGE

1st Edition November 1968

© The University Mathematical Laboratory, Cambridge

PREFACE

The Cambridge multiple-access system has been developed for the Titan (prototype Atlas 2) computer by a team under the direction of Professor M.V. Wilkes at the University Mathematical Laboratory, Cambridge. The system is a direct development of the Titan supervisor, for which the early work was carried out in conjunction with I.C.T. Ltd., (now ICL). The multiple-access system has been in regular service in the Laboratory since March 1967.

The team principally concerned with the development of the system was led by R.M. Needham and included the following: D.F. Hartley, B. Landy, A.G. Fraser and M.J.T. Guy. In addition, the following have been actively involved in the work at some time: D.W. Barron (who was in charge of the early work referred to above), D. Barton, S.R. Bourne, J.M.H. Hunter, C.A. Lang, P.R. Radford and J.C. Viner. Mrs. J. Skilling has given valuable assistance with the production and maintenance of the documentation on which this manual is based. Finally it is important to acknowledge the staff members and students of the Mathematical Laboratory and other users of the Titan computer who have provided many valuable suggestions and written a number of important commands in the system.

The work of setting up the system was largely supported by the Science Research Council.

CONTENTS

		Page
INTRODUCTION		1
1. CONSOLES		
1.1	Connection	5
1.2	Basic character set	6
1.3	Escaped characters	7
1.4	The BREAK key	9
1.5	Unexpected responses	9
1.6	Multi-console working	12
1.7	Satellite computer	12
2. ACCESS TO THE SYSTEM		
2.1	Control	13
2.2	User identifiers and passwords	13
2.3	Project numbers	14
2.4	File owners	15
2.5	Access modes	15
2.6	Operating styles	16
2.7	Shifts	17
2.8	Off-line use of the system	17
2.9	Accounting	18
3. THE MANAGEMENT SYSTEM		
3.1	The log-in parameters	20
3.2	Logging in	21
3.3	Logging out	27
3.4	Warnings and automatic log out	27
3.5	Restart effects	29
4. THE COMMAND SYSTEM		
4.1	General details	31
4.2	Format of commands	32
4.3	Standard environment and execution	38
4.4	Exit from a command	39
4.5	The dump mechanism	40
4.6	The COMMAND command	41
4.7	Use of the command system off-line	43

		Page
5.	THE FILING SYSTEM	
5.1	Introduction	44
5.2	File titles	45
5.3	Operations on files	46
5.4	Preferred versions	48
5.5	Classes of files and space allocations	49
5.6	Security	51
5.7	Relationships	53
5.8	Interlocks	58
5.9	Archive mechanism	58
5.10	File dumping and recovery	59
5.11	Directory inspection	60
5.12	File errors	61
6.	PRIVATE COMMANDS	
6.1	Command filing	62
6.2	Entry and exit	63
6.3	Parameter handling	64
6.4	Security and errors	65
7.	COMMAND SUMMARY	
7.1	Categories of commands	67
7.2	Summaries of important commands	67
Appendix 1	Teletype/internal code tables	78
Appendix 2	Filing fault numbers	82
Appendix 3	The 1026 extracode	83
Appendix 4	File directory details	84
Appendix 5	The management algorithm	87
Appendix 6	Index of commands	90
Index		91

INTRODUCTION

The CAMBRIDGE MULTIPLE-ACCESS SYSTEM runs on the Titan computer (prototype Atlas 2) in the University Mathematical Laboratory, Cambridge, and also on the Atlas 2 computer at the Mintech Atlas Centre, Madingley Road, Cambridge. The two computers differ somewhat in configuration, but both have 128K words (48 bit) of core storage, and a Data Products disc file; on the University computer, the disc file has a capacity of about 8 million words, and on the Mintech computer it has a capacity of about 24 million words. Both computers have magnetic-tape decks, and paper-tape input and output devices. There is also provision for punched cards. Communication with the operators is effected by a special typewriter on the operators' desk. Each computer is provided with a Calcomp plotter, and with a satellite computer (PDP7 and PDP9 respectively) which incorporates a graphical display.

The Cambridge multiple-access system allows authorised users to have access to the system either on line from a console or off line, via punched paper tape or cards, in the more conventional manner. A user is not tied to one mode of access, and may, indeed, work partly from a console and partly off line on the same problem, using whichever mode of access seems most suitable for the stage he has reached. In addition to being able to run jobs from a console and have the results come straight back, it is also possible to initiate from a console a job which will then run exactly as if it had been presented off line.

The user is provided with a general-purpose filing system in which he can store material in files which he refers to by names of his own choosing. He will expect to be given a limited ration of space on the disc for his files and, possibly, a more generous ration on magnetic tape in what is known as the archive system. There are elaborate provisions whereby the user may make his files available to others, and whereby users may work in groups.

A system of passwords prevents unauthorised persons having access to the system, and provides for the security of users' files. Users are given an allocation of time in each shift, together with an overall allocation for the month. If a user exceeds one of these allocations, he is given a warning message; however, unless the system is fully loaded and another user wishes to log in, he is allowed to continue in what is known as overtime. Otherwise, he is requested to finish off and log out, and, if he does not do so within a reasonable time, he is logged out automatically. Users are also controlled as to the amount of core space they may use, and the amount of output they may create. There is an elaborate system of priority which provides, among other things, for a user being given a small allocation of time at high priority, and a larger allocation at low priority; it also enables users to be grouped for priority purposes into divisions.

The computer runs under the control of a supervisor, which is a program whose essential parts are permanently resident in core storage and which is responsible for all the complex administration and scheduling that is necessary. The amount of core space permanently allocated to the supervisor is only 15K, but the supervisor can, and does, make use of additional space when it is not being used for any other purpose. An important function of the supervisor is buffering input and output. Input and output devices pass information, under the control of the supervisor, to or from buffer areas on the disc; this mechanism is generally known as the well. Information in the well is held in the form of documents, and users' programs communicate with these documents and not directly with input and output devices. Such buffering is used in order that the mechanical input and output devices may run at their own speed without holding up the processor, and also in order that a number of programs can use the same line printer, for example, without their outputs getting mixed up. A further important function of the supervisor is that of code translation, which is rendered necessary by the fact that different peripheral devices work with different character codes.

This manual is intended to provide all information needed by the on-line user, although he will, of course, also need to refer to other manuals describing the various programming languages offered. In particular, if he wishes to program in assembly language, he will need to refer to the Titan Machine-Code Programming Manual. This latter manual also contains information about how job descriptions are written for the purpose of presenting jobs to the system off line. For the convenience of readers of the present manual, an explanation is given below of such terms as streams, backing-store devices, and extracodes.

When a user writes programs, he thinks of all input and output information as going to, and coming from, streams. Only later, when the program is presented for running, are these streams associated with particular input and output devices. In the case of an off-line job this is done in the job description; in the case of an on-line job it is done by the user typing information as part of the command initiating the job.

256 input streams, and 256 output streams are available, numbered in each case from 0 to 255. Output streams are distinguished by the letter O; input streams can be marked either P or D, the intention being that the former are program streams and the latter data streams, although the distinction between program and data is not, of course, an absolute one. Certain commands and sub-systems test the type of a stream and distinguish between P and D.

There are various actions that the programmer cannot cause to be performed directly, but must make a request to the supervisor to perform them for him. Among these are input and output, and the creating, closing, or deleting of streams. In a high-level language the fact that a call is being made on the supervisor is concealed from the user, but if the user is

writing in assembly language, then he will need to make use of extracodes. These are machine-code instructions, and correspond to what, in other systems, are known as supervisor calls. Many of the things done in advance by means of a job description can be done at run time by means of an extracode. At various places in this manual, the explicit forms of extracodes are given, and the reader who is not familiar with machine-code programming for the Atlas 2 should disregard them.

An input stream can be set up in a number of ways. In an off-line job it can be connected to a document that has originated from a slow peripheral (such as a card reader), or it can be connected to a magnetic tape. An on-line job can have an input stream connected to a console. Any job can also have input streams connected to files and to documents that have been previously stored in the well through an output stream. Similarly, an output stream may be set up as a document in the well and destined for an output peripheral (line printer, paper-tape punch etc.), or it may be connected directly to a magnetic tape (off line only) or to a console (on line only).

It is possible to create an output stream and to leave it pending; what happens to the stream is determined when it is closed, either by an extracode or by an on-line user typing an appropriate command.

There are also backing-store devices which can be regarded by the programmer as abstract entities; when a program comes to be run, a backing-store device can be declared to be a magnetic tape, a file, or temporary working space on the disc. Backing-store devices are used for holding binary information which is accessed in blocks of 512 words. This is compatible with the system of recording on Atlas magnetic tapes, namely in pre-addressed blocks of 512 words each. The declaring of a backing-store device to be a magnetic tape is only permitted to off-line jobs.

A typical job running on the Cambridge multiple-access system will pass through a number of phases. One such phase might be compiling, another loading, and a third running. Each phase is treated as a self-contained task, and a separate allocation of storage space is made by the supervisor. The execution of a command issued by an on-line user is organised as one or more such phases.

This manual is primarily intended as a reference document for users of the on-line consoles, and is organised in the following manner. Chapter 1 gives information pertinent to the consoles themselves. This information is relevant however the consoles are used and is independent of any particular command, programming language, or other sub-system. Chapter 2 is concerned with the various ways in which users may access the system, and with the resources that are available, while Chapter 3 gives details of how users begin and end on-line sessions. Chapter 4 describes the command language used from the consoles, and Chapter 5 describes the filing system for retaining users' information on the disc file. Chapter 6 gives details

of how users can create their own private commands. Summaries of the more important system commands are given in Chapter 7. It is not practical for this document to give complete specifications of all commands currently available; from time to time, certain commands may be modified, and new commands provided. Full and up to date specifications of system commands are available to users of the system in the separate document entitled THE CAMBRIDGE MULTIPLE-ACCESS SYSTEM, COMMAND SPECIFICATIONS, which is updated as necessary. Finally, various system details are given in a number of Appendices.

CHAPTER 1 - CONSOLES

The system will accommodate various types of console, although at present those in use are all Teletype Model 33. Some of the particulars given in this chapter are peculiar to this type of machine.

1.1 Connection

Consoles have two connections, one to the mains supply and one via a jack plug to a line leading to the multiplexor of the computer. For consoles in the vicinity of the installation these lines are connected directly to the multiplexor; more remote consoles may be connected via Post Office Modems and the public telephone network.

The majority of consoles are controlled by a 3-position rotary switch on the front, the positions being labelled LINE, OFF, and LOCAL.

LINE switches the machine on and connects it to the multiplexor or modem.

OFF switches the machine off.

LOCAL switches the machine on but does not connect it to the multiplexor or modem.

On other machines the OFF position is omitted, and the motor is controlled by a button. When the console is not being used, it should be switched off, and it is good practice to switch off at the mains as well.

If the console is connected to a modem, then connection to the computer is achieved as follows. Lift the handset and dial the appropriate number. When a member of the operating staff answers, ask to be connected to the system; if the operator confirms that the system is available, press the button on the telephone instrument labelled DATA and leave the handset off the instrument. When it is required to disconnect the call at the end of a session, press the button labelled TELE and replace the handset.

Each line (and hence each connected console) has a number by which it is known to the system and which depends on the configuration of connections and plugs to the multiplexor. This console number is used to report console activity to the operators and for internal purposes. In the ordinary way it is of no concern to the user; however, it may be used by the administration to restrict access in various ways, for example, to allow only the consoles in the computer room to be used, or to restrict use of the Titan-PDP7 data link to the user of a console located near the satellite computer. A console number therefore depends upon the location of the console; although it is normally fixed, the number associated with a particular location may be changed from time to time for engineering reasons.

1.2 Basic character set

The Teletype is essentially a 3-shift machine, the two extra shifts being obtained by holding down the SHIFT or the CTRL key while some other key is pressed. Some of the combinations are mechanically locked, and others do not print but exist for various purposes connected with the original application of the device in telecommunications. In general these combinations will send signals to the multiplexor which are treated as error conditions by the system. The SHIFT key must be pressed to obtain the upper character when two characters are engraved on the key, and the CTRL key must be used to obtain the control function when this also is engraved on the upper part of a key. A control function is denoted by two or more letters, and of those available only TAB and BELL are expected to be useful to the majority of users. On some older machines, the characters [ ] and \, although available, are not engraved on the keys; in this case, these characters are obtained in the SHIFT set by pressing K, M and L respectively.

In normal use, the user of a Teletype terminates each line of typing by striking the RETURN key, which has the immediate effect of moving the type head to the start of the line; in response to this, the system will automatically issue a LINE FEED character, thus moving up the paper by one line. This serves two purposes. First, it saves the user from having to strike an extra key; and secondly, if the automatic line feed fails to appear it indicates a failure in the system (see 1.5.6). Alternatively, the ALTMODE key can be used to terminate a line of input without actually repositioning the typehead of the console. This is sometimes useful in certain types of interactive programs where it is desired to maintain a particular layout on the printed page.

Since all normal character handling in the computer is conventionally done in terms of a standard internal code, irrespective of the origin or destination of the information, the software associated with the consoles converts all information from a console from its local code to internal code, and vice versa for information sent from the computer. In the ordinary way, of course, a user of the system need not be aware of these code translations. This is only necessary when writing character handling programs, or sometimes when dealing with information which has originated from, or is destined for, peripheral devices whose character sets differ from those of the Teletype.

A table of characters and their internal code equivalents is given in Appendix 1. This includes certain characters which can only be typed by indirect means as described in 1.3. As a general rule, any character which is not directly available on the Teletype, or not available via the mechanism of 1.3, cannot be input via the Teletype; while if the system is asked to output on the Teletype a character which is not directly available, then a '.' symbol is produced in its place.

1.3 Escaped characters

The character '@' indicates an escape; on its own it has no direct effect on the input matter, but is used as a means to extend the character set of the Teletype. The escape is recognised specially by the system, and causes it and the character following to be interpreted as a single character, known as an escaped character. This mechanism provides:

(a) useful characters not directly available on the Teletype,

(b) local editing operations on console input,

(c) signals to the program associated with the console.

1.3.1 Extra characters

These characters are provided by conventional substitution, and the following are typical examples:

@@ produces internal code outer set 3.2 (i.e. this is how one enters an '@' in the input).

@V produces internal code outer set 2.7 (a vertical bar).

For a specification of which characters are available reference should be made to Appendix 1.

1.3.2 Local editing

The following escaped characters can be used to correct typing errors; they apply only to the line currently being typed, and are always available whenever information is being typed one line at a time, irrespectively of what command or program is in use.

@C deletes the previous character typed in the current line.

@C@C deletes the two previous characters; and so on.

@L deletes the current line, and the system responds with a RETURN and LINE FEED.

More precise statements expressed in terms of the record structure common to all input and output, are as follows:

@C deletes the most recently typed character (or escaped character) in the current record; it has no effect if the current record is empty. Emptiness of the record can arise in three ways:

(a) Nothing has been typed since the last character which represents a carriage control character.

(b) The record has been exhausted 'from the front' by a program which takes characters as soon as they

have been typed (note that the majority of programs do not use this facility).

(c) The record has been exhausted 'from behind' by repeated use of @C.

@L deletes the whole current record (if any). Note that this is not necessarily the same as the current console line if records are terminated by carriage control characters other than RETURN or LINE FEED; for example, if ALTMODE is used. @R is an alternative to @L and has the same effect.

1.3.3 Signalling characters

@Q, @I, and @M are used to send signals to the command or program currently associated with the console. They are conventionally used as follows:

@Q	quit	The command is abandoned.
@I	interrupt	An interrupt is sent to the command, and the effect depends upon the command itself; if the command does not cater for interruptions, then the effect may be as for @M.
@M	monitor	The command is abandoned, but in certain cases will first produce a diagnostic indicating its current state.

In all cases, the system initially responds with a RETURN and LINE FEED.

The particular details of how the signalling mechanisms operate are as follows:

@Q produces fault 93. This is an unusual kind of fault in that it never causes entry to the system monitor. It may cause entry to a private monitor if one has been set; otherwise, the end program sequence is entered (this is equivalent to the 1117 extracode) whence control is returned to the command system.

@I produces fault 65, which is a conventional delayed fault and may be trapped by the program if desired. When such a trap occurs, the first information b-register is set to the (actual - see 1.6) console number. User and system programs can be organised so that @I provides a general interrupt mechanism.

@M produces fault 94, and is similar to @Q except that system monitor is entered if appropriate in the normal manner. @M provides a means for obtaining the standard monitor output and stopping a user program. In system programs, it is usual for @M to have the same effect as @Q.

1.3.4 Other characters

@N is ignored and gives no response; it is the recommended way of cancelling a mistyped @.

@ RETURN gives a LINE FEED response, but is otherwise ignored; this enables a new line of typing to be started without terminating the current line as far as input to the computer is concerned.

Other characters typed after @ are treated as errors; they cause the system to respond with a further @ symbol to enable the erroneous escaped character to be corrected or cancelled.

1.4 The BREAK key

Communication with the associated program using the signalling characters as described in 1.3.3 is not possible while the console is being used for output, that is, when it is typing under computer control. It is necessary in this case first to stop the output so that the required signal can be given. This is done by pressing the BREAK key and holding it down for about a second; the effect is to set up a state recognised by the multiplexor software which causes the following actions to take place:

(a) output (if any) is stopped, and the contents of the output buffer are thrown away (up to 120 characters);

(b) the system gives the response:

*** @

After this, the next character typed by the user is treated as an escaped character, and it is possible either to send a signal as described in 1.3.3, or to have the break ignored (apart from loss of output) as described in 1.3.4. Thus BREAK is equivalent to both stopping output and typing @.

1.5 Unexpected responses

This section lists certain conditions (mainly due to errors of one sort or another) that give rise to output on the console. Almost all the messages begin with *** to attract the user's attention to what is normally an unexpected response.

1.5.1 Input errors

Typing a character or control character that has no assigned meaning, i.e. does not appear in the tables of Appendix 1, causes the response

*** FAULT CHARACTER

This is followed by the last few characters received so that the user may continue typing correctly. The fault character itself has no effect on the

input material.

An error in transmission from the Teletype to the computer may either give rise to the above response, or, more commonly, will produce

*** ERROR

As before, this is followed by the last few characters correctly received. This condition will also occur if the computer is stopped for a short time (by operator intervention or otherwise) with the result that some typed characters are lost; in these circumstances the user is able to repeat these characters after the response.

1.5.2 Input buffer congestion

The input buffer maintained by the system can hold up to 120 characters. In exceptional circumstances, software delays can cause this buffer to become full, giving rise to the response

*** TOO MUCH INPUT
*** WAIT

When the backlog is cleared sufficiently to enable typing to proceed, the system types

*** CONTINUE AFTER

followed by the last few characters received. Alternatively, if the user terminates or interrupts the current command before the CONTINUE AFTER response is given (by typing @Q, @I or @M),the input record is automatically terminated (with the carriage control character value 0.0 - see Appendix 1).

If the input buffer becomes full when it contains no complete record (i.e. there is a very long line), the record is terminated automatically (with carriage control character value 0.0), and the system responds

*** LINE TERMINATED

There is no CONTINUE AFTER message in this case.

1.5.3 Warnings

The following messages may be typed by the system; they are concerned with the management system and are described in 3.4.

*** PRIORITY WARNING - PLEASE FINISH OFF
*** OPERATOR WARNING - PLEASE FINISH OFF
*** COMP EXHAUSTED - OVERTIME SET
*** EXEC EXHAUSTED - OVERTIME SET
*** COMP EXHAUSTED - PLEASE FINISH OFF

The typing of these messages has no direct effect other than to interrupt the user's current input or output.

1.5.4 Messages

Public and personal messages may be sent to the user. A public message will give rise to

*** MESSAGE -

followed by the text of the message itself.

A personal message is indicated by

*** MESSAGE FOR YOU

Either of these responses can only occur while the user is in command status (see 4.1); the text of a private message is obtained by using the MESSAGE command.

1.5.5 Starts and restarts

Whenever the system is completely initialised at the start of each day or after a serious failure, the characters

SSSSSSSS

appear on each switched-on console. Any user who was previously logged in must, if he is to continue, log in again, and no trace will be retained of his previous activity (other than changes to files).

Whenever the system is restarted after a hardware or software failure the characters

RRRRRRRR

appear on each switched-on console. A user who was previously logged in (and who has not received a PRIORITY warning - see 3.4) will be able to continue working when the restart is complete; in general this takes a minute or so. No trace will have been retained of the user's previous activity (other than changes to files).

Off-line jobs (including those created by the user from a console - see 2.8) are dealt with as follows. On a start, all jobs in the system are abandoned. On a restart, jobs which were in the process of execution when the failure occurred are begun again; otherwise, any job which had not started execution or had finished execution is not affected.

1.5.6 Other phenomena

If the system fails to respond to RETURN with LINE FEED, or if each character typed by the user is immediately typed back by the system, then either the system is not in service or it has just failed. In the latter case, the service will either continue after a few moments delay (typically the user receives an ERROR response as described in 1.5.1), or a restart will occur (see 1.5.5). If the failure is such that service will be sus-

pended for more than a few minutes, the operating staff will cause a suitable message to be typed on all switched-on consoles.

Of course, situations can arise which are not covered by the above. If in doubt, the user should contact the administration.

1.6 Multi-console working

Under certain circumstances it is possible for a user working at a console to have other consoles associated with his program. This permits certain types of interactive application where several people communicate (independently) with a single program. For these purposes, each logged-in user may be connected to up to 8 consoles. These 8 consoles are referred to by logical console numbers 0 to 7, and facilities are available which enable an actual console to be connected as one of these. Ordinarily a user is connected to only one console, which is logical console 0 although he does not need to be aware of this.

Extra consoles can be connected only if the user is working in EXPENSIVE mode (see 2.5); the connections are made by using the SIGNAL command.

1.7 Satellite computer

A PDP7 (or PDP9) computer with type 340 display is connected to the system via a special data link. The data link can be used by suitably authorised users with the effect that the satellite computer and its associated equipment becomes a sophisticated graphics terminal. The PDP and SIGNAL commands are provided to establish and use the link.

CHAPTER 2 - ACCESS TO THE SYSTEM

2.1 Control

All access to the system is controlled by the system itself from information supplied to it by the administrative staff. The system controls the amount of file space, core space, and machine time allocated to each individual user, and, according to the mode of operation of the machine at any time, controls the type of access that he may have.

Before he can use the system, a prospective user must first obtain from the administration an allocation of system resources (e.g. space for files, computation and execution time allocations for on-line working, etc.). He will be given a user identifier by which he will be known to the system, and a project number. This having been done the user can gain access to the system at one of the consoles by logging himself in. The log-in procedure, which is part of the management system and is described in detail in Chapter 3, requires the user to state his identifier and project number and to give his password. The management system also enables the user to demand any special resources or other privileges that might be permitted to him.

The user identifier also serves the purpose of identifying the user's files if he has been allocated file space by the administration. A user who has not been allocated file space is not permitted to own files, although he may have access to the files of other users. Not all owners of files are individual users of the system; files may belong to groups of users, to system libraries, etc. Thus, although it is normal for each user to have a user identifier and a project number, and to own files in the system, it is sometimes necessary to distinguish between the following:

(a) an individual user,

(b) a project number which has an allocation of time associated with it,

(c) a file owner.

The following sections go into these concepts more fully.

2.2 User identifiers and passwords

A user identifier generally consists of a sequence of one or more letters (commonly the user's initials) possibly followed by an integer; its total length is 8 or less characters, and it must be typed without spaces. Associated with each user identifier is a password, which must be quoted on each occasion that the user logs in. The password provides a security mechanism which enables a user's files and time allocations to be safeguarded against accidental or malicious misuse by others. On the other hand, a user may be permitted to have controlled access to another user's files (see 5.6).

When a user is first given access to the system by the administrative staff, his password is set to a null value (i.e. no characters) and he can log in by simply typing a RETURN when asked for his password. The first time he uses the system, a new user should set himself a password by means of the PASSWORD command. This command may be used to change the password at any subsequent time at the user's convenience for security or other reasons. If a user forgets his password, he can have it reset to the null value upon application to the administrative staff.

Since the system retains information about user's passwords in a system file, it is possible (but unlikely) that, if the files on the disc are lost and dumped copies retrieved from magnetic tape, a password may be 'backed up' to a previous version if it had been changed recently by the user.

2.3 Project numbers

An individual user will be issued with one or more project numbers, which relate to the particular project or projects approved by the administration. It is also possible for the administration to permit several users who are working as a group to have access to the system under the same project number.

Associated with each project number are allocations of time. The amount of time used under any project is measured by the system in two separate ways: COMP time is the amount of central processor time used, and EXEC time is a measure of how long a console is in use. The terms COMP and EXEC are defined more precisely in 2.9. Each project is allocated separate amounts of COMP and EXEC for each of the working shifts (see 2.7). Such allocations are made on a monthly or on a once-for-all basis, a monthly allocation being renewed automatically on the first working day of each month. Each allocation is made at a particular priority level (see 3.1), and in some cases a project may have several allocations at different priority levels in the same shift.

The administrative staff can apply various controls to each allocation of time given to a project. These controls enable the system to regulate the manner and the rate at which time is used. For example, the controls may limit how much COMP and EXEC is to be available in any one shift; they specify which particular user or users are permitted to use an allocation, and they are used to determine which access modes (see 2.5) are available to these users depending on the current operating style (see 2.6). The controls applied to a project may be different for the different allocations of time made to it.

The appropriate allocation is debited by the amounts of COMP and EXEC used whenever a user logs out. The user can find out how his project's allocations stand by using the PROJECT command. If he is unable to log in to do this, he can obtain the information from the administrative staff.

Each project number is allocated to one of a number of divisions. Divisions form administrative groupings of users, and they enable the management system to distribute the load of logged-in users amongst these groups when the system is heavily loaded. In particular, this helps the management system to decide which user should be asked to log out when a higher priority user wishes to log in. Details of this mechanism are given in Appendix 5.

2.4 File owners

A file owner is normally identified by a sequence of up to 8 letters and digits. If a file owner is a user this identifier is the same as his user identifier.

The files belonging to a file owner are normally held on the disc store, and there is an automatic mechanism for storing copies of files on magnetic tape, both for protection against disc failure and for archive purposes. The space available for files is limited by the capacity of the disc store, and each file owner is given an allocation of file space by the administrative staff. Full details of the filing system are given in Chapter 5.

2.5 Access mode

In any multiple-access system which provides general programming facilities, certain levels of interaction can be identified, according to the way in which a user interacts with the computer while it is processing his work. For the purposes of the Cambridge system, it is convenient to consider three distinct levels of interaction:

(a) Remote job entry

The user submits a complete job to the system from a console (see 2.8). It is not possible to communicate with such a job while it is running, although after the job has finished the user may inspect any results that have been sent to files.

(b) Non-interactive commands

At this level, the user can give a command which is executed almost immediately. During execution, the program executing the command does not communicate with the console; but once completed any suitable output from the command is immediately typed on the console. The set of commands is entirely open-ended, and at the present time provides facilities ranging from simple housekeeping operations to the servicing of requests for program compilation and program execution in a variety of programming languages.

(c) Interactive commands

A program called by an interactive command communicates with

the user's console during execution; the program may cause information to be typed directly on the console and may receive input from it, so that the writer can design the program to be as interactive as he chooses.

All these techniques are in principle available in the system, although certain restrictions are placed on users in order that system efficiency can be maintained. In particular, interactive commands are generally more expensive in terms of system resources than the other techniques. To enable as many users as possible to have suitable facilities while maintaining a high level of machine efficiency, the system provides various access modes. The access mode of a user is determined when he logs in, and controls what kind of work he may do at the different levels of interaction.

The majority of users ordinarily work in what is called NORMAL mode. In this mode, a user may use the remote job entry feature and any non-interactive command; in addition he may use certain system interactive commands such as INPUT, EDIT etc. EXPENSIVE mode, which is generally only available to certain users, permits any interactive command to be called. Even then, the amount of core space that may be used for such commands is limited; this limit is a multiple of 4K words, and one refers to EXPENSIVE 4K mode, EXPENSIVE 8K mode, etc.

EDIT mode is provided for the situation in which a user requires only a restricted set of facilities. In this mode, it is possible to use only the remote job entry feature and certain system commands such as INPUT, EDIT and those concerned with file operations.

SPECIAL mode is provided for system maintenance and administrative operations, and enables members of the administrative staff to log in without disturbing other users; in other respects it is identical to NORMAL mode.

The access mode required by a user is specified when he logs in; once set, it cannot be changed without logging out and then logging in again.

2.6 Operating styles

The system has various styles of operation, which can be selected by the computer operating staff. These styles control who may log in at any time and in which access mode. A typical user may be allowed NORMAL or EDIT mode in some operating styles, EXPENSIVE mode in other operating styles, and no access in the remainder. Which modes are available in which operating styles is determined by the administrative staff separately for each project. The following table lists the operating styles together with the access modes available to ordinary users.

STYLE	PURPOSE	TYPICAL ACCESS
0	all users normal access	NORMAL, EDIT
1	as 0, + EXPENSIVE mode for PDP7 link	NORMAL, EDIT
2	as 0, + EXPENSIVE mode for demonstrations	NORMAL, EDIT
3	unrestricted	EXPENSIVE, NORMAL, EDIT
4	spare	none
5	administration only	none
6	as 0, + EXPENSIVE mode for software development	NORMAL, EDIT
7	software development only	none
8+n	as style n for computer room consoles only	none
16	no on-line service	none

2.7 Shifts

The time of a normal working day is divided into 3 shifts as defined in the table below. Amounts of COMP and EXEC allocated to projects are organised on a shift basis so that the administrative staff can distribute the available resources over different times of the day. An extra shift is provided for unscheduled times (e.g. parts of the weekend).

SHIFT	TIME OF DAY
0	09.00 to 17.00 (daytime)
1	17.00 to 24.00 (evening)
2	00.00 to 09.00 (night)
3	unscheduled (week-end)

2.8 Off-line use of the system

In addition to working on line, a user may initiate jobs to be run off line. There are two ways in which this can be done.

(a) The RUNJOB command provides a remote job entry mechanism. This command enables an on-line user to inject a job directly into the off-line job stream. Off-line jobs are processed in parallel with multiple-access working,

and, according to the current load on the system, any job initiated by an on-line user may or may not be completed while the user remains logged in. A user is not permitted to have more than one job that has been created in this way in the system at any one time.

(b) Jobs can be submitted to the off-line computing service in the normal way. This method must be used if a job includes input from paper tape or cards.

In either case, an off-line job is presented to the system in the form of a job description, the details of which are outside the scope of this manual (see Titan Machine-Code Programming Manual, Chapter 10).

All the facilities of the filing system (described in Chapter 5) are available to off-line jobs, so that a user may create and edit files from a console, initiate an off-line job (which uses the files and produces results in the form of other files), and subsequently examine the results from a console.

In addition to files, all sub-systems (editors, compilers, commands etc.) are available off line, including the command system itself (see 4.7).

2.9 Accounting

The precise definitions of COMP and EXEC are as follows:

COMP is the actual amount of central processor time required to execute the programs called for by a user's commands, and includes certain system overheads.

EXEC is the amount of COMP plus the time that the command system and programs called by a user's commands have to wait for input from the console or for backing-store transfers to be completed; since waiting for console input is typically very much the larger of these, EXEC gives a measure of how long the console is in use less any time that the user must wait for system delays. The latter are largely due to the fact that the computer is shared simultaneously by several on-line users and off-line jobs.

COMP and EXEC are the only measures of computer time known to the management system, and the on-line control of users is done entirely in terms of them. Retrospective accounting for use of the machine time is, however, done in terms of an entirely different measure that takes account of the amount of core space occupied by each phase of a user's program as well as of the processor time (COMP) that it consumes. The effect is that a user is debited with more computer time if his program occupies a large amount of core space than it is if his program occupies a small amount of core space. The formula used is as follows:

For each phase of the job, the user is charged

$$\frac{t+5}{20}(s+a) \quad \text{seconds}$$

where

t = COMP used (rounded up to nearest 0.08 seconds)

s = store used (in units of 1024 words, rounded up to nearest half unit)

a = 4 if $0 < s \leq 8$

12 if $8 < s \leq 16$

24 if $16 < s \leq 32$

36 if $32 < s$

Both on-line and off-line jobs are accounted for by this mechanism. In these terms, a single logged-in on-line session is a job, while a command gives rise to one or more phases.

The intention in introducing this measure of computer time was to make it clear to users that programs that occupy a great deal of core space put a relatively heavy load on the system, and to encourage them to keep their programs short. The considerations that led to this decision apply in any multi-programmed computer system in which core storage is allocated dynamically to programs of differing size. The above formula is not intended to be an exact or even approximate expression of the value of core space in a cost accounting sense; it is designed rather to give users a feel for what they are doing in terms of load on the system.

The management system for on-line users and the accounting algorithm just described were developed independently and it has not, unfortunately, proved possible to bring them together. There would be everything to be said for using the above measure instead of COMP for all purposes; unfortunately, the implementation difficulties involved in doing so make it likely that the present arrangements will continue in force for the lifetime of the system.

CHAPTER 3 - THE MANAGEMENT SYSTEM

The purpose of the management system is to regulate the use of the multiple-access system by individuals according to the resources of computation time and execution time allocated, and the controls upon their use set by the administrative staff. The user comes into contact with the management system through the log-in procedure, which determines if and how he may use the system when he starts to use a console, and through the warning procedure, which may indicate that the user must stop work to enable another to have access. Both of these are described in detail in this chapter.

3.1 The log-in parameters

When a user logs in, the management system checks that he has the necessary resources allocated to him and determines whether he can be accommodated. There are also various options that may be specified by the user, such as which access mode he wishes to work in, and these affect the decisions as to whether and how he may have access.

PRIORITY

In any particular shift, a user, or more accurately, a project has an allocation of COMP and EXEC at some priority level. There are four priority levels called LOW, MEDIUM, HIGH, and TOP. When the system is fully loaded, the priority levels are used to determine whether a user may log in, and if so, who should be warned off. Often a user's allocations are all at MEDIUM priority; it is possible, however, for some users to have different priorities in different shifts, and to have more than one allocation in the same shift at different priorities.

MODE

A user must determine which access mode he requires when he logs in (see 2.5). Which mode is permitted to him depends both on the controls associated with the project number being used, and on the number and modes of other users currently logged in.

LIMSTORE

LIMSTORE is the term given to a parameter which controls the maximum amount of core store permitted to the user at any one time. Usually this is set to 12K, but may in principle be set to any value between 2K and some maximum which is set by the operating staff and which may vary from time to time; usually the maximum is 32K, but it may be as high as 40K.

LIMOUT

The LIMOUT parameter controls the amount of disc space available for input/output streams and backing-store devices which are not files (for full

details see the Machine-Code Manual, Section 10.4.6.4). LIMOUT may be set to any number of blocks up to 1024. In default, it is set to 200 blocks of disc, which is sufficient for most purposes.

COMP and EXEC

COMP and EXEC parameters specify upper limits to the amounts of computation time and execution time respectively that may be used during the current logged-in session. The amounts of COMP and EXEC that may be granted depend on

(a) the amount of unused time allocated to the project in the current shift and at the requested priority,

(b) the controls associated with this allocation, and

(c) the upper limits set by the operating staff.

If the user exceeds the COMP limit while he is logged in, he is either put into OVERTIME status (see below) or given a warning; if the EXEC limit is exceeded, the user will be put into OVERTIME status.

OVERTIME

If an allocation (COMP or EXEC) given to a project becomes exhausted, or if certain controls associated with this allocation are violated (for example, if more than a certain amount of time has been used in the current shift), then the management system may permit the user to log in, or continue to remain logged in, in OVERTIME status, provided that the system does not become overloaded.

The main effect of OVERTIME status is to remove the priority of the allocation being used, with the result that the user is more likely to receive a priority warning (see 3.4.2) than otherwise. In other words, the user may log in or continue to work until the system becomes fully loaded and some other user wishes to have access. Note that time used in OVERTIME status is debited against the project's total allocation in the normal way.

3.2 Logging in

3.2.1 The log-in procedure

To log in, a user types a RETURN on a free console. If the system is in operation and generally available, it will respond with

LOGGING IN

followed after a short pause by

USER

After this the user types his user identifier, terminated by a RETURN.

There is a further response of

PROJECT

after which the user types the project number under which he desires to work, again terminated by a RETURN.

If it is possible for the user to log in, that is if the user's resources and the current load on the system do not rule out ab initio the possibility that he will be allowed to log in, then the system proceeds to ask for the user's password. This it does by responding with a row of certain superimposed characters and leaving the type head at the start of the line. The user then types his password on top of these characters, and terminates it by RETURN; the result is sufficient to disguise the password from prying eyes. Having checked the password, the system may then type a 'message of the day', which will contain useful and timely information about the running of the system.

At this stage, the system will usually respond with

OPTIONS?

This indicates that the system is able to permit the user to log in with the standard parameter settings. If the user now types a RETURN, then he will be logged in with these parameter settings. The standard settings are as follows:

MEDIUM PRIORITY
NORMAL MODE
LIMSTORE 12K
LIMOUT 200
COMP 5 MINS
EXEC 60 MINS

By typing a RETURN after the OPTIONS response, the user completes the log-in procedure and the system responds with the message

PASSWORD LAST USED time - date

which provides a security check on when the password was last used or changed. After a pause, the length of which depends on the load on the system, there will be a further response of

STARTED time - date

followed by

READY

The user is now in command status (see 4.1) and can begin work.

The above describes the log-in procedure as it applies in its simplest form; that is, it assumes that the user requires the standard parameter settings and that the system can accommodate him in this way.

If there is no way that the user can be logged in (that is, if the system is so loaded that no possible settings of the parameters will enable the user to have access), the system responds with a message giving the primary reason why this is so (e.g. SYSTEM FULL) and then proceeds to give the following message:

```
STYLE x SHIFT y
NO ACCESS time - date
```

This message is always given when the log-in procedure does not lead to the user being logged in. It is usual in such cases for the system to omit password checking and the typing of the message of the day.

In some circumstances, typically when the project has an allocation of resources at a priority higher than MEDIUM, the system may find that the user can be logged in but only with parameter settings different from the standard. In this case, before the word OPTIONS is typed, the system responds with the primary reason why straight-forward logging in is not possible, followed by a proposal of parameter settings which will allow logging-in. For example, it may type

```
SYSTEM FULL
PROPOSE EDIT MODE
OPTIONS?
```

The user may then accept the proposal by typing a RETURN, or alternatively he may attempt to modify the proposal by typing different parameter settings after the word OPTIONS. These must all appear on the same line, terminated by RETURN, and the following formats are permitted:

LOW PRIORITY MEDIUM PRIORITY HIGH PRIORITY TOP PRIORITY	(the word PRIORITY may be omitted)
OVERTIME	
SPECIAL MODE EDIT MODE NORMAL MODE EXPENSIVE MODE	(the word MODE may be omitted)
LIMSTORE nK	(n is in units of 1024 words; K is optional)
LIMOUT n	
COMP n MINS EXEC n MINS	(the word MINS may be omitted)

When EXPENSIVE mode is specified, the required limit of core space for interactively obeyed commands (see 4.1) may also be given by typing nK after the word EXPENSIVE. The number n is rounded up to the nearest multiple of 4K; if no number is typed, a limit of 4K is assumed. The limit that will be accepted is determined by the controls associated with the project and by the current load of other users on the system.

Whether or not the system has proposed any changes to what he originally requested, the user may re-specify parameter settings at any time when the system responds with OPTIONS. Thus, in the above example, if the user decides that he would rather try to exert his HIGH priority (assuming that he has it) and have NORMAL mode, than accept EDIT mode as proposed, he would type

HIGH PRIORITY NORMAL MODE

The system reconsiders the parameters, and either

(a) logs in the user, if this is possible, or

(b) if it is not possible, reports back with the reason, makes a further proposal, and types OPTIONS again.

In this way, the user may converse with the system, until either agreement is reached, or the user decides not to accept what the system can provide. In the latter case, the user must conclude the log-in procedure by typing the word FINISH after OPTIONS.

To save time, certain standard parameter settings are automatically made unless the user explicitly states otherwise. These are as follows:

(1) If a change is made in the priority proposed or requested, then COMP and EXEC are set to the standard values of 5 MINS and 60 MINS respectively, and it is assumed that OVERTIME status is not required even if previously proposed.

(2) If the only reason why the system cannot accept the required parameter settings is that the COMP and EXEC it can allow are different from those requested (the EXEC being greater than 4 minutes), the system will respond with an amended COMP and/or EXEC time and, if the user has previously had the opportunity to change the parameters, it will log him in without OPTIONS being further requested.

There is one important point to note. Since there may be several users attempting to log in simultaneously, the system may be inconsistent in its proposals, particularly if the user is slow in taking action. For example, the system may have indicated that log in is possible, but by the time the user has responded to OPTIONS the situation may have changed so that it is no longer possible.

The message of the day may indicate that there is more information available if the user wishes to have it. In this case, the information is obtained by typing the word MORE, either with or instead of, parameter settings. In either case, the system will type the appropriate information and then type OPTIONS again.

3.2.2 Errors and other conditions

Responses given by the system during the log-in procedure should be self-explanatory but the following notes should clarify some of the more abnormal circumstances:

(1) In response to the initial RETURN:

SYSTEM FULL	the computer is temporarily overloaded; try again after a short interval.
SESSION STARTING	the system will be available shortly.
SESSION ENDING	service is about to be concluded.
ON-LINE SYSTEM NOT AVAILABLE	the computer is operating but nct for general on-line work.

If there is no response other than a further RETURN, then the system is not available.

(2) If the user types an unknown project number, or if his user identifier (which may have been typed or transmitted incorrectly) is incompatible with the project number quoted, then a suitable message is typed, and the system asks for both the user identifier and the project number again.

(3) If the user mistypes his password, the system responds INCORRECT PASSWORD and asks for it again.

(4) An individual user is only permitted access on one console at any time. An attempt to log in twice will give rise to

USER ALREADY LOGGED IN AT CONSOLE m

(5) The system reports any errors found in information typed after OPTIONS by repeating the erroneous information in the form

information UNKNOWN

All information before the error is accepted,but the rest of the line is ignored. The word OPTIONS is then typed again to allcw the error to be corrected.

(6) @I can be used to interrupt the message of the day and the MORE information. At other times @I causes the system to act as follows:

(a) after USER or PROJECT, to type USER again.

(b) after the password has been requested, to request the password again.

(c) after OPTIONS, to type OPTIONS again.

@Q and @M cause the log-in procedure to be abandoned.

(7) The procedure may be terminated by system errors, which are reported as

SYSTEM ERROR (n)

If such happenings persist, the operating staff should be informed and told which value of n is being given.

(8) If the user is particularly slow in replying to the system's requests, log in may be terminated with the message:

TIME EXPIRED

(9) If the system responds with

SYSTEM OVERFULL

then this means that there is a temporary overload in the number of users actually logged in, although some of them have received warnings (see 3.4.2) and will be logging out shortly. In this case, try again in a few minutes.

(10) If a RETURN is typed at any time after the system has typed PASSWORD LAST USED etc. and before it has typed STARTED etc., the system responds with

WAITING

3.2.3 Alternative procedures

The experienced user can save a small amount of his time by one or other of the procedures given below.

The user identifier can be followed on the same line by the project number (separated by a space). In this case, the response PROJECT is omitted.

The project number can be followed on the same line by a space and then by one or more parameter settings (as normally typed after the response OPTIONS). In this case, if log in is found to be possible, OPTIONS is omitted after the password has been checked; the message of the day (if any) is given, and the user is logged in without further formalities. Note that this will happen even if the COMP and EXEC values are modified by the system: the user will be told that this has happened, but will not be given the chance to modify his parameters in the light of it.

Similarly, typing the word GO after the project number, indicates that standard parameter settings are required, and that OPTIONS is to be dispensed with if log in is possible.

3.2.4 Message sending

Public or private messages can be sent from any console by means of the SIGNAL command, but of course the user must be properly logged in to do so. For the convenience of the operating staff it is possible to send messages as part of the log-in procedure, thus avoiding the need to go through the full procedure. This facility is available only at console number 0 which is located in the operating area of the computer.

At this console, any user may type the word SIGNAL after OPTIONS followed by any of the usual parameters for the SIGNAL command. The management system enters the SIGNAL command directly, and the log-in procedure is abandoned; subsequently, the console is available to begin the log-in procedure again in the normal manner.

3.3 Logging out

To log out, the user types the command FINISH at any time that he is in command status. The system will respond with

FINISHED time - date

On log out the system debits the amount of COMP and EXEC used during the session against the appropriate allocation of the project. If the shift had changed while the user was logged in, then the times are debited roughly pro-rata against the allocations in the two shifts. The console is then released from the system, and after a brief interval may be used again to enter the log-in procedure as described in 3.2.1. Note that, until the system is ready for the console to be re-used, typing a RETURN produces the response

WAITING

3.4 Warnings and automatic log out

There are a number of situations in which a logged-in user may be automatically removed from the system. In all but extreme cases this is done in such a manner that he has sufficient warning to be able to straighten out his affairs without loss of information.

3.4.1 Automatic OVERTIME

If more COMP or EXEC time is used than was allowed for by the parameter settings when he logged in, a user is permitted to continue work in OVERTIME status. The setting of OVERTIME status occurs automatically; the user receives a suitable message to say that this has been done, but his current work is otherwise unaffected. Note that when in OVERTIME status a user is likely to receive a priority warning if the load on the system increases (see 3.4.2). The details of the automatic OVERTIME mechanism are as follows:

(a) EXEC time exhausted
The message *** EXEC EXHAUSTED - OVERTIME SET is typed on the console. The user is set in OVERTIME status and is given enough extra time to ensure that he has at least 1 minute of COMP, and 1 hour of EXEC. If he remains logged in for a further hour the process may be repeated.

(b) COMP time exhausted
If the user is already in OVERTIME status, then he will be given a time warning (see 3.4.2); otherwise the message *** COMP EXHAUSTED - OVERTIME SET is typed and the system proceeds as for (a) above.

Thus a user may continue to work indefinitely as long as the number of users wishing to log in remains below the maximum that can be accepted, and as long as he never uses more than 1 minute of COMP to each 1 hour of EXEC. It has been necessary to impose this last condition to avoid serious waste of computer time if a user abandons a console and leaves a program in an infinite loop.

3.4.2 Warnings

Except in the very special circumstances mentioned in 3.4.3, a user is never automatically logged out without first being given a warning that this is about to happen. The purpose of the warning is to indicate to the user that he must immediately begin to tidy up his work by filing all information that he wishes to have preserved. The different types of warning and how they can arise are specified below. They all have the same effect; namely, a suitable message is typed and the user is permitted a further 30 seconds of COMP and 3 minutes of EXEC only. Failure of the user to log out before either of these times become exhausted causes automatic log out (see 3.4.3). The messages and their causes are as follows:

(a) *** PRIORITY WARNING - PLEASE FINISH OFF
The system is fully loaded and a new user is logging in with higher priority than that of the present user; note that, if the present user is in OVERTIME status, his own priority has no significance. The mechanism that decides which user actually receives a priority warning in these circumstances is described in Appendix 5.

(b) *** OPERATOR WARNING - PLEASE FINISH OFF
The operating staff require the user to log out. Usually this only happens at the end of a computing service session.

(c) *** COMP EXHAUSTED - PLEASE FINISH OFF
The user's allocation of COMP time has been exhausted in OVERTIME status (see 3.4.1(b)).

Note that a user cannot be warned more than once, unless a restart occurs (see 3.5).

3.4.3 Automatic log out

To avoid waste of console time through a user abandoning a console without logging out, no user is permitted to leave his console unattended for more than about 10 minutes at a time in command status. If, in this status, a user fails to type a command for 5 minutes, the system will type the message:

TO REMAIN ON-LINE YOU SHOULD TYPE A CARRIAGE RETURN WITHIN 5 MINUTES

leaving the type head at the end of this line. If the user then fails to type a RETURN within a further 5 minutes, the system assumes that the user is no longer present, and automatically enters the log-out procedure.

The system automatically logs out a user if COMP or EXEC becomes exhausted and he had previously been given a warning. The system types

*** COMP EXHAUSTED

or

*** EXEC EXHAUSTED

as the case may be, followed by the normal response of FINISHED etc. Apart from files, nothing is preserved.

In exceptional circumstances, the operating staff may need to log out a user without warning. This causes the system to type

*** OPERATOR STOPPED

and the effect is otherwise as above.

A user is also logged out automatically if a certain type of hardware failure occurs in the console communications equipment. In these cases, it is not possible to inform the user directly; the console immediately repeats each character typed, and when the system next responds correctly, it is by the log-in procedure starting in response to a RETURN being typed.

3.5 Restart effects

As mentioned in 1.5.5, the system types RRRRRRRR on each switched-on console when there is a restart. This occurs whether or not a user is logged in at the console. How this affects a user depends on circumstances as follows:

(a) If the user was in the middle of the log-in procedure, he must start again from the beginning.

(b) If the user had logged in (i.e. the system had at least responded with PASSWORD LAST USED etc.), then he remains logged in, and must wait until he receives the response

RESTARTED time - date

which will be followed by

TO REMAIN ON-LINE YOU SHOULD TYPE A CARRIAGE RETURN WITHIN 5 MINUTES

If the user fails to type a RETURN within 5 minutes, the system assumes that he has abandoned the console (the restart may have followed a prolonged breakdown), and the log-out procedure is entered automatically. If the user types a RETURN in time, then the system further responds with

READY

to indicate command status.

(c) If the user had received a priority warning previous to the restart, he must try to log in again if he wishes to do further work. In the case of other types of warning, he remains logged in as in (b) above, but he can expect to receive a further warning in due course. In a similar way, if the user has been automatically set into OVERTIME status as described in 3.4.1, he is restarted in normal status, but can expect to be put into OVERTIME status in a short while.

CHAPTER 4 - THE COMMAND SYSTEM

The main purpose of the command system is to recognise individual commands given by users from consoles, and to arrange for the necessary programs to be executed. The commands provide the usual means of access to the filing system, to the editor, and, in general, to any programs available through the on-line system. The command system can also be used off line for jobs which are organised as a sequence of elementary commands of the same sort as might be used on line. This chapter is mainly concerned with the normal use of the command system (interactively) by a logged-in user; other uses are described in 4.6 and 4.7.

4.1 General details

After a user has completed the log-in procedure (see 3.2) or after there has been a restart (see 3.5), the system eventually responds with READY. This indicates that the user is in command status, and that he may type any command known to the system together with any necessary parameters. The actions initiated by commands range from simple operations to those performed by complicated sub-systems, such as compilers, etc.; in addition, there may be private commands belonging to individual users.

When a user has typed a command, the system arranges for the appropriate program to be loaded into core storage and executed. On completion of this program, control returns to the command system which, after performing certain housekeeping operations, types the word READY indicating that the user is now in command status once more. The FINISH command, which logs out the user, is an exception to this rule.

Commands fall into two categories: interactive commands which communicate directly with the console, and non-interactive commands which are executed without such direct communication. Certain system commands are normally obeyed interactively, and these are listed in Appendix 6. All other commands (including private commands) are obeyed non-interactively, unless the user is in EXPENSIVE mode; in the latter case, the user may choose how he wishes the command to be obeyed.

Communication to and from a command, particularly a non-interactive command, is often achieved by means of input and output streams, and occasionally through backing-store devices. Streams and backing-store devices may be used internally by a command, and may be set up externally when the command is typed. Many commands are written so as to operate on an input stream or streams, and to produce results in the form of an output stream or streams. In particular, the following conventions are often observed. Input, other than simple parameters, is taken from a particular input stream (input stream 1), which is generally referred to as the current document. Output from a command is conventionally accumulated on output stream 0; on completion of the command, control returns to the command system which then converts this document into a particular input

stream (input stream number 252), and types a copy on the console. This stream is generally referred to as the results document. Many commands operate in this manner, namely by taking the current document as input and producing a results document as output; other commands may produce a new current document. An exception to these conventions is made for interactive commands. In this case output stream number 0 is connected to the console so that the output will be typed directly as it is produced and not retained as a results document.

4.2 Format of commands

In its simplest form, a command is typed simply as a command name; for example

INPUT	calls the INPUT command.

Other commands may be written with parameters following the command name in what is known as an argument string. The form taken by parameters depends upon the command itself; frequently these are file titles and in many cases they may be optionally specified. Thus

EDIT	calls EDIT to edit the current document
EDIT /PROG/A1	calls EDIT to edit the file /PROG/A1 which it first opens as the current document.

In many situations it is convenient for the user to request the setting up of various input/output streams or backing-store devices at the time that a command is given. For example, he may be using some of the more advanced features of the EDIT command, or perhaps calling one of his own programs which communicates with specific streams or devices. These, together with certain other standard actions such as specifying a limit to the computation time for the command, can be performed by the command system itself immediately before the command is entered. There is a standard notation for specifying these actions known as an environment declaration. This consists of a sequence of one or more items (for details see 4.2.2), enclosed in round brackets, and written after the command name and before the argument string (if any). For example

AUTOCODE (D2 /ABC) /PROG/A1	calls the Autocode system to compile and run a program from the file /PROG/A1; in addition, the file /ABC is opened as input stream 2 to be read as data by the program.

The general form of a command consists of the following separate parts, any of which may be absent:

command name; environment declaration; argument string.

A command is terminated by a RETURN character; but note that a RETURN in round brackets in either the environment declaration or the argument string does not terminate the command. Full details of the separate parts are given in the following sections.

4.2.1 Command names

In general, a command name identifies a file in the filing system which contains the appropriate program. The exceptions to this rule are certain important commands whose instructions are held with the command system itself; there is, however, no difference in the way that they are specified. Typically, a command name is written as a sequence of letters and digits (it must not contain a space). Alternatively, a command name may be a file title (see 5.2), and since a file title always includes at least one '/' symbol, there is no confusion between the two forms.

Extra SPACE and TAB characters preceding or following a command name are ignored. An empty command name signifies 'no action' other than that denoted by an environment declaration if present. A command cannot consist solely of an argument string for obvious reasons.

4.2.2 Environment declarations

The environment declaration provides a standard notation through which the user can request certain actions to be performed and can specify certain parameters that are commonly required when a command is called. The actions include the opening of files as input/output streams or backing-store devices, while the parameters may specify the amounts of core storage and COMP time required. Both of these can be thought of as providing an environment for the program called by the command in the sense that they may specify not only how the program is to be handled by the operating system, but also which files of information are to be used by it on this occasion. For many simple commands, an environment declaration is unnecessary; its main use arises in commands that call for user programs to be executed, since the latter are commonly written to communicate with particular input/output streams or backing-store devices. When writing programs, the user should observe the following conventions in connection with the use of streams and backing-store devices. Input and output streams numbered 200 and above and backing-store devices numbered 120 and above may be used internally by any sub-system. In particular the command system itself uses certain of these.

An environment declaration is always enclosed in round brackets, and generally consists of a number of directives. The various parts of each directive are usually words, letters, integers, file titles, or other symbols; SPACE, TAB or RETURN may be typed before or after any part as desired, and one of these characters must be used to separate any two parts

which might otherwise be ambiguous (i.e. between file titles and words, letters or digits, and between words and letters). In the following descriptions the term 'backing-store device' is sometimes abbreviated to BSD. The following types of directive are permitted; they may be given in any order with several of each type if desired.

Input stream directives

Pn s_1 s_2 ...

Dn s_1 s_2 ...

Both forms deal with input streams according to the specifiers s_1, s_2, ...; s_1 specifies what is to be done with input stream n, s_2 refers to input stream n+1, and so on. In each case, if the action is to create an input stream, the stream is made P-type or D-type according to the initial letter. The possible specifiers are given below, together with the actions that are taken.

file title	input stream created from the specified file.
*	input stream closed.
=	input stream left in its current state.
:Pm :Dm	input stream renamed from input stream m which is rewound if necessary (if the two stream numbers are the same, the input stream is merely rewound).
:Om	input stream renamed from output stream m.
p/q	input stream created from BSD p at block q (the BSD must be a magnetic tape, and therefore this form is only available off line).
H	input stream created from material <u>on subsequent</u> lines, which is sometimes known as a <u>here</u> document. There are two alternatives: (a) if H is followed on the same line by some character (ignoring SPACE or TAB), the material is terminated by a line consisting solely of that character; (b) if H is followed only by a RETURN, the system responds with *** READY; the material is typed after this and terminated by @Q.

	Z	equivalent to :I252 (input stream 252 is conventionally known as the results document).

NOTE: If the first directive of an environment declaration is to begin P1,..., then the characters P1 may be omitted.

Output stream directives

On s_1 s_2 ...

This form deals with output streams according to the specifiers s_1, s_2, ...; s_1 specifies what is to be done to output stream n, s_2 refers to output stream n+1, and so on. Each specifier takes one of the forms given below, and may be preceded where appropriate by an integer giving a limit to the size (in blocks) of the output stream to be created; if the integer is omitted, a limit of 10 blocks is normally assumed (see Limit directive).

	file title	output stream connected to the specified file.
	*	output stream closed (and deleted if 'pending').
	=	output stream left in its current state.
	p/q	output stream created to BSD p at block q (the BSD must be a magnetic tape, and therefore this form is only available off line).
	PRINTER	output stream created and designated as going to a printer.
	PENDING	output stream created and designated as pending.
	PUNCH	output stream created and designated as going to a 7-track paper-tape punch.

Creates output stream n as pending; that is, the stream will be sent to disc and its ultimate destination (if any) is to be decided later. The integer m specifies a limit to the size (in blocks) of the stream; if it is omitted, a limit of 10 blocks is normally assumed (see Limit directive).

Backing-store device directives

Nn s_1 s_2 ...

Wn s_1 s_2 ...

Rn s_1 s_2 ...

Each of these forms deals with backing-store devices according to the specifiers s_1, s_2,...; s_1 specifies what is to be done to BSD n, s_2 refers to BSD n+1, and so on. Where relevant, the initial letter indicates how the BSD is to be opened:

N a new file is created,

W an existing file is opened for updating,

R an existing file is opened for reading only.

Each specifier takes one of the forms given below, and may be preceded where appropriate by an integer giving a limit to the size (in blocks) of the BSD (which is relevant in the case of a new file or an existing file which is to be extended); if the integer is omitted, a limit of 10 blocks is normally assumed (see Limit directive).

file title	BSD created as the specified file.
*	BSD closed.
=	BSD not affected.

Tn m

Creates BSD n as temporary working space on the disc (the space remains available only so long as the BSD exists). The integer m specifies a limit to the size (in blocks) of the BSD; if omitted, a limit of 10 blocks is normally assumed (see Limit directive).

Limit directive

Kn

Sets an assumed limit of n blocks for the sizes of any output streams and BSD's created by directives which follow this directive.

Time directive

Cn

The operating system provides a local clock which can be set to cause an interrupt after a specified amount of COMP time has been used. The command system uses the clock to limit the amount of machine time used by each command; this is particularly useful when non-interaction commands are used to call untested programs. A time directive sets the local clock to expire after n seconds; if one is omitted, a command is permitted 30 seconds. Note that the program called by a command may always reset the local clock; this is done, for example, in certain system commands where a time limit is irrelevant.

Store directive

Sn

Specifies that the program called by the command is to be loaded into n words of storage. This directive is relevant to certain commands (public and private) only; in the case of public commands the command specification indicates when it is necessary. If the directive is omitted, a store size of 4096 words is assumed.

Interaction directive

X
Xn

These forms indicate that the current command is to be obeyed interactively. They are unnecessary for those public commands which are normally obeyed interactively, and are only relevant to users in EXPENSIVE mode. The integer n (if specified) gives the store size for the command. (In fact, Xn is equivalent to X Sn).

Synonyms

Instead of single letters, words may be typed in full as follows.

C	COMP
D	DATA
H	HERE
K	OLIMIT
N	NEWFILE
O	OUTPUT
P	PROGRAM
Q	PENDING
R	READFILE
S	STORE
W	OLDFILE
Z	RESULTS

4.2.3 Argument string

The argument string may be any sequence of characters, subject only to the following rules: (1) initial SPACE or TAB characters are ignored, and (2) the string cannot start with '(' unless an environment declaration is also present (otherwise the string would be taken as an environment declaration). The string may include newlines, as long as these occur in at least one level of round brackets.

The argument string is used to specify parameters to the command being called, and hence its precise format depends on the command itself. It is

not normally necessary to know how the argument string is handled, except perhaps that input stream number 253 is used. Further details are provided in Chapter 6 for the benefit of users who require to write their own commands.

4.2.4 Errors

If a command contains a format error, if the name of the command does not correspond to a valid file, or if the environment declaration requires some action which is not possible or permitted, then the system responds with a suitable error message. Certain types of system failure also give rise to error messages, and in some cases it is probably worth while repeating the command. If the failures persist, the user should inform the operating staff.

After a failure, the system types READY and thus returns to command status. Anything typed by the user before READY will be ignored. If a user does not complete a command but types @Q, @I or @M instead, the system responds with READY and returns the user to command status.

4.3 Standard environment and execution

In addition to giving effect to any environment declaration or argument string, the following standard actions are taken, depending on the type of the command and the mode of the user.

(1) If the user is in EDIT mode, the system checks that the command name is one of those indicated in Appendix 6. If the command is not permitted, a suitable message is given, and the environment declaration (if present) is ignored.

(2) Input stream 0 and output stream 0 are dealt with as follows:

   (a) if the command is to be obeyed interactively (that is, if it is one of the standard interactive commands (indicated in Appendix 6), or if the user is in EXPENSIVE mode and has specified interaction in the environment), then the streams are connected directly to the user's console;

   (b) otherwise, the command is to be obeyed non-interactively; input stream 0 is closed, and output stream 0 is created as a 'printer' stream with an output limit of 10 blocks. (In normal circumstances this stream will not actually be sent to the printer - see 4.4).

If, however, either stream is explicitly mentioned in the environment declaration, then the specified action takes precedence over the preceding rules.

(3) Finally, the command system arranges to load the program of the command into the appropriate amount of core space, and enters this program at address 0. If the user is in EXPENSIVE mode and the command has been called interactively, the system checks that the amount of core space used is within the limit for interactive working specified when the user logged in.

During execution of a command, the following restrictions are applied by the system:

(a) If the command is being obeyed non-interactively, it is forbidden to connect input/output streams directly to a console.

(b) If the user is in EXPENSIVE mode and the command is being obeyed interactively, the amount of core store in use while input or output streams are connected to a console must be within the limit specified for interactive working. (For example,if the user is in EXPENSIVE 8K mode, any phase of the program called by the command that communicates directly with a console must not be more than 8K words in size.)

4.4 Exit from a command

When the execution of a command is completed, control returns to the command system which performs certain standard actions (detailed below) before returning the user to command status. A command is completed either through the execution of an 1117 end-program extracode (or its equivalent in higher level languages), or through an error which gives rise to system monitor action. In either case, no core image of the program is preserved unless the user has explicitly requested this to be done in advance (see 4.5).

The following standard actions are performed by the command system on exit from a command and before returning the user to command status.

(1) The following input streams are closed:

stream numbers 200 to 254,
any stream (other than input stream 1) connected to a file,
any stream connected to the user's console.

(2) The following output streams are closed:

stream numbers 200 to 255,
any stream connected to a file,
any stream connected to the user's console.

(3) The following backing-store devices are closed:

BSD's 121 to 126,
any BSD connected to a file.

(4) A results document may be produced in the following manner:

If output stream 0 is designated as going to a peripheral or is pending and has been used at all by the command, then this stream is re-named as input stream 252 (D-type). Then, if this has been done, the system copies this stream to the console (equivalent to the command TYPE RESULTS). This typing can be interrupted in the usual manner by use of the BREAK key.

(5) The system finally types READY to denote the return to command status.

Note that these actions are taken if the command is called in the normal manner from the user's console. The system takes different actions if input stream number 255 exists on exit from a command; this is the case when either the COMMAND command is used (see 4.6) or the command system is used by an off-line job (see 4.7). It is important therefore that the user does not himself use input stream number 255.

4.5 The dump mechanism

Unlike certain other multiple-access systems, the Cambridge system does not automatically preserve a core image of a program when a command terminates either naturally or as the result of an error. Instead, the operating system provides a mechanism which enables the core image of a program to be dumped, together with values of the accumulator, b-registers, trap settings, etc., if this is requested in advance by the user. A dump is made in such a way that the user may subsequently interrogate and alter the core image and associated information by use of the PATCH command. It is also possible, in certain circumstances outlined below, for the execution of a dumped program to be resumed or restarted.

Dumping must be specified in advance either by use of the SETDUMP command or by an 1115 extracode obeyed in a program. In either case, dumping is specified to occur in the file with the title /DUMP//FFRR.T, or on a backing-store device of the user's choosing. These facilities do not actually cause a dump to take place, but instead they indicate that any program subsequently called is to be dumped in the manner specified either if a fault arises when the program is executed causing the system monitor mechanism to be entered, or whenever the program obeys an 1116 extracode. The SETDUMP command (or 1115 extracode) may also be used to inhibit and then subsequently to permit the dump mechanism; in this way it is possible to arrange that dumping does not occur if faults arise in certain programs or parts of programs. In addition, dumping is automatically inhibited during system commands or any other command that is specified to the user as 'load only'.

The format of a dumped program is as follows. The first block contains values of the accumulator, b-registers, trap settings, etc., together with a bootstrap routine which enables the program to be reloaded into the correct amount of storage and re-entered. Subsequent blocks contain the core image of the program. The PATCH command (see 7.2) provides the user with a number of facilities for interrogating the information in a dump; he may also alter this information and then resume or restart the program. The user must beware, however, that resumption of a program may not always lead to useful results if the dump was made as a result of a fault. It is also important to note that the state of any input/output stream or backing-store device is not preserved by the dump mechanism. Full details are given in the TITAN MACHINE-CODE PROGRAMMING MANUAL, Section 9.12.

4.6 The COMMAND command

4.6.1 Purpose

A user may prepare a sequence of commands (which he may store in a file) and then subsequently request this sequence to be executed as though it were a single command by calling the COMMAND command. If the command sequence is stored in a file, the argument string given after the name COMMAND must include the title of the file containing the command sequence; it may possibly be followed by a parameter list (see below). If the file title is omitted, the current document is assumed to contain the command sequence. The parameter list enables the user to specify one or more strings of characters that are to be inserted into the command sequence before the commands are executed. Each character string must be preceded by a $ character, and any string may include spaces if desired. For example

```
COMMAND /ABC/3 $623$/DATA/7$XYZ
```

The positions at which these strings are to be inserted within the command sequence are marked by the characters $n, where n is a single non-zero decimal digit, indicating that the nth string in the parameter list is to be inserted in this place. Thus, if the command sequence called by the above example contains

```
FILE $2
```

this will be executed as

```
FILE /DATA/7
```

Also in the command sequence, $ may occur followed by some other character; this denotes that subsequent parameter position markers are introduced by this character and not by $. The character § may be used instead of $ both in the parameter list and in the command sequence.

The execution of a command sequence takes place in the following manner.

If the command sequence contains no interactive commands, then the next response received by the user will be the typing of a results document containing the accumulated (concatenated) outputs from the commands as produced by them in turn (unless, of course, some of the commands themselves have otherwise dealt with output stream 0); this is followed by READY denoting the return to command status. If the command sequence contains an interactive command, then any results accumulated from previous non-interactive commands are first typed before that command is entered.

If the command system detects an error in a command or in an environment declaration, an appropriate error message is given in the usual manner, and the remainder of the command sequence is abandoned. The system produces and types the results document (which will end with the error message), then types READY, and returns to command status. This action is also taken if any command finishes as a result of an error detected by the supervisor; that is if execution is ended via a system or private monitor sequence. It is important to note that this does not apply to any other form of error detected by the program of the command itself, unless the program indicates such an error to the command system in the manner described in 6.2.

4.6.2 Method

For any but simple uses of the COMMAND command, it is useful to know some details of how the mechanism is organised.

The document containing the command sequence first has substituted in it any parameters that have been specified, and is then renamed as input stream 255; if input stream 255 already exists, then the system assumes that COMMAND has been used within a sequence called by COMMAND, and it merges the new sequence with the old to obtain the effect of nested COMMAND commands. This is all that the COMMAND command actually does, and the remainder is accomplished through the way that the command system behaves if it finds that input stream 255 exists. For this reason it is important that this stream is never used for any other purpose.

If, on exit from any command, the command system finds that input stream 255 exists, then the actions are as follows:

(1) Certain input/output streams and backing-store devices are closed as described in 4.4.

(2) The next command is read from input stream 255 and decoded. If this is a non-interactive command, then it is entered in the normal way with an output stream created and designated to the printer, if such a stream does not already exist. If the command is interactive, then, before it is entered, the system types the results (if any) of the previous command(s) that may have accumulated on output stream 0 in the same manner as described in 4.4.

When input stream 255 becomes exhausted, or if an error is detected in decoding or executing a command as described earlier, then the stream is closed and the command system proceeds in the normal manner for command exit; to the user, these actions indicate a return to command status.

4.7 Use of the command system off line

The command system can be used by off-line jobs for the convenience of being able to specify a sequence of operations, but, of course, without the benefits of interaction. The usual technique is to write the word COMMAND as the system name in the job description, with the command sequence specified as either input stream 254 (the NOTES stream of a job description) or input stream 0. Alternatively, COMMAND can be called later in the job by some other system (for example, by the E3 editor).

On the initial entry, the system performs two special actions:

(a) A marker is set in the operating system's working space for the job indicating that the 1117 end-program extracode and the system monitor mechanism are to return control to the command system, rather than terminate the job.

(b) If input stream 254 exists, the document is renamed as input stream 255.

Otherwise, the command system operates off line in a similar manner to the COMMAND command (see 4.6), with the following exceptions:

(1) No commands can be interactive, and, unless special action is taken by the commands themselves to deal with output stream 0, the results of the commands as accumulated on this stream will be printed at the completion of the job.

(2) Whenever input stream 255 becomes exhausted, or if an error is detected in a command, as described in 4.6.1, this stream is closed, and input stream 0 used instead.

(3) When input stream 0 becomes exhausted, or if an error is detected in a command as described in 4.6.1, the system automatically calls the FINISH command, which terminates the job.

Note also that, when the command system is used off line, all commands read from either input stream 255 or input stream 0 are line-imaged to remove erases etc. using the normal conventions for 7-track Flexowriter paper tapes. This treatment also applies to environment declarations but not to input streams imbedded in them.

CHAPTER 5 - THE FILING SYSTEM

5.1 Introduction

The filing system provides a mechanism for the storage of information belonging to users; this information can consist of programs in source language, programs in binary, data etc.; users are in fact free to store in a file any information they choose.

The system is available equally for both on-line and off-line use; the files themselves are created as named documents which contain information in an arbitrary format. Files are ordinarily held on the disc file store connected to the computer, and as such are readily accessible to users at the consoles and to programs within the machine. A file is handled internally and stored on the disc as a sequence of blocks (1 block = 512 words = 4096 6-bit characters); certain further pieces of information associated with the file such as its name and other details are kept separately on the disc by the system, and they are not directly accessible to the owner.

The most usual kind of file contains a string of characters (for example, a program in source language or some data); this is known as a character file. To create such a file from the console the user can adopt the following procedure. First he uses the INPUT command to type in the required information; this command creates a document in the correct format. Secondly, he uses the FILE command to store this document in a newly created file, at the same time assigning it a title by which the file can be identified. Subsequently, he may change the contents of this file by use of the EDIT command, and may arrange for copies of it to be typed on the console or sent to the line printer. If the file contains a program, the user may call the appropriate compiler to compile this program by giving a suitable command and specifying the title of the file to be used.

In addition to files containing information in character form, there are also block files. These contain information in binary form, so that, for example, a compiled and assembled program may be filed away in such a manner that it is available for execution through a simple loading procedure.

The filing system is common to all users and the amount of space for files on the disc is limited; it is therefore necessary to limit the amount of file space available to each user. It is also essential that a user's files should be protected from unauthorised access by other users. Accordingly, the filing system contains administrative devices for file space allocation and file security and provides a wide range of optional facilities which may be used to control access to files in a variety of ways.

In addition to files being stored on the disc, the system automatically retains copies of certain files on system magnetic tapes. This mechanism provides two facilities: first, it guards against loss of files as a result of system failures, and secondly it permits infrequently used files to be

stored on magnetic tape instead of on the disc, increasing the total capacity of the filing system.

5.2 File titles

Each file in the system is identified by a unique file title. A file title is made up of three separate components, the first of which is the identifier of the file owner (see 2.4). The second and third components are names chosen by the owner to suit his own convenience, and it is a common practice to choose them so that the second represents a 'file name' while the third is a 'version name'; either of these components may be empty. The three components are separated from one another by '/' characters, and each component may contain up to eight characters chosen from the following character set:

Letters A to Z
Digits 0 to 9
Symbols . + -

The following are examples of titles of files belonging to the user AGF210:

AGF210/PROG/A1
AGF210/DATA

in the latter case, the third component is empty. Since users typically access their own files, the first component of a file title may be omitted when it is being used by its owner. Thus, the user AGF210 may type the above titles as

/PROG/A1
/DATA

In situations where a user repeatedly refers to files belonging to a different file owner, he can use the SET command to specify that the identifier of that owner is understood when the first component is omitted.

A complete file title includes a fourth component. This component, called the file status, is in two parts: the first part consists of four letters known as the access status of the file; the second part consists of a '.' character followed by a single letter which specifies the class of the file. The access status is primarily concerned with the access to the file by other users and is described in 5.6. The class specifies how the file is to be treated; for example, it may be a permanent or a temporary one. Details are given in 5.5. A file status need only be typed by the user when he creates the file, and even then, either or both parts may be omitted if standard settings of the access status or class are required. The following are examples of file titles written with a fourth component:

AGF210/PROG/A1/.P
/DATA//FFRN.T

Each component of a file name may contain 8 characters in addition to the '/' characters. There is, however, an unfortunate restriction in the implementation that prevents the system from accepting more than 28 characters altogether (including '/' characters) in the name of a file. This means that if the second and third components are long, it may not be possible to include the status letters as part of the file name in a FILE command. Obviously, the best thing is to avoid long names if status letters have to be set explicitly; however, if this is not convenient, the STATUS command can be used to set the status letters (see 7.2). The rule for terminating a title depends on the context in which it is being used; typically this might be a space or a newline. The particular rules followed by the filing system itself when decoding titles that are presented to it are as follows:

(a) Shifts and erase characters are ignored.

(b) Each component is terminated by a '/' character or after the 8th printing character.

(c) The internal code character 0.7 is treated as a printing character for the purposes of (b) but is otherwise ignored.

(d) The title is terminated by an extra '/' character, or by internal code character 0.0.

For the extracodes which control the basic operations on files see 5.3.

5.3 Operations on files

When new information is to be stored in the filing system, a file must first be created and given a file title. At the same time, the file is opened to enable the information to be transferred to it. When the transfer is completed, the file will then be closed. To access the file at some later time, a user must again open the file indicating as he does so how the information in the file is to be read and/or updated; once more, when access is complete the file must be closed. Ultimately, when the information that it contains is no longer required, the file may be deleted. The operations of creating, opening, transferring information, closing and deleting are available to users in a number of ways, and all of them are provided by commands that can be typed on a console. For example, the command

```
FILE AGF210/PROG/A1
```

creates a file with the specified title (automatically deleting any existing file with the same title), opens it, and transfers the contents of the current document to the file. All filing system operations are provided by extracodes, so that any manipulation on files may be programmed by the user in assembly language, and in any language which provides the necessary subroutines.

There are two basic methods of accessing files corresponding to charac-

ter files and block files. In the case of a character file, it is usual for the file to be stored in the conventional stream format that is used for all input/output handling in the system. Such a file is created and opened as an output stream and characters are transferred to it by the usual output functions; the operating system packs the characters into blocks which it then transfers to the disc. Conversely, the file is read by opening it as an input stream and then reading information from the stream with the usual input functions; again, the operating system attends to the details of disc transfers and unpacking operations. This is the usual technique for handling files of characters, and it is used for this purpose by all standard commands and sub-systems.

The alternative method of accessing a file enables each block to be handled as a single unit, so that information may be arranged within these blocks in any way. In this case, a file is opened (if necessary, first created) as a backing-store device, and then accessed by writing or reading blocks in the same manner as if the file were a standard pre-address Atlas magnetic tape. This method is commonly used for filing compiled and assembled programs in binary.

Note that the filing system itself does not distinguish between the two types of file, and it is possible to treat as a block file one that has been created as a character file and vice versa, although it is unusual to do so.

The following are examples of filing operations that are provided in various parts of the system.

(a) Commands

TYPE AGF210/PROG/A1	opens the character file, and types a copy on the console.
COPY AGF210/TEST /PROG/A2	makes a new file with the second title as a copy of the file with the first title.
DELETE AGF210/DATA/XYZ	deletes the file.

(b) Environment declarations (see 4.2.2)

P1 /PROG/A1	opens the file as input stream 1 (P-type)
O3 /RESULTS	creates and opens the file as output stream 3
N1 AGF210/PROG/BIN	creates and opens the file as BSD 1
W5 /A12/3.4	opens an existing file as BSD 5
R3 AGF210/BCD/1	opens an existing file as BSD 3 for reading only.

(c) Job descriptions

```
INPUT P1 FROM FILE (/PROG/A1)
OUTPUT 3 TO FILE (/RESULTS)
TAPE 1 NEW FILE (AGF210/PROG/BIN)
TAPE 5 FILE (/A12/3.4)
TAPE 3 READ FILE (AGF210/BCD/1)
```

} These are similar to the examples in (b) above

(d) Extracodes

1030	opens a file as an input stream
1040	opens a file as an output stream
1021	opens a file as a backing-store device
1026	provides a wide range of filing system operations.

A summary of the facilities provided by the 1026 extracode is given in Appendix 3, and full specifications are available in the COMMAND SPECIFICATIONS document. Details of other extracodes are given in the Titan Machine-Code Programming Manual.

Note that there are two ways in which a file may be deleted. First, a file may be deleted explicitly by use of the appropriate command (or extracode). Secondly, a file is automatically deleted when a new file is created with the same title as the old one; in this case, the file is said to be implicitly deleted.

5.4 Preferred versions

A user may frequently have several files which contain different versions of the same information. In this case, he may adopt the convention of making the second component of each file title identical, and using the third component to denote a version name or number. If this convention is followed, the prefer mechanism provides a convenient way of referencing a chosen one of these files. This mechanism is as follows:

For any group of files whose titles each have the same name as their second component, one of them is defined to be the preferred version. This is either that file which has been explicitly nominated by use of the PREFER command or the 1026/6.4 extracode, or, if no such action has been taken, the most recently created or updated file in the group. A user may specify that the preferred version of a group of files is to be used by typing a '*' character in place of the third component of the title.

For example, the command

PREFER AGF210/PROG/A3

marks this file as being the preferred version of files with titles of the form AGF210/PROG/... Subsequent to this command, whenever a user refers to a file with the title

AGF210/PROG/*

then the above file is used until such time as either this file is deleted, or another version is explicitly preferred. Alternatively, the above form refers to the most recently created or updated file of the group. Note that in defining the term 'most recently' the system does not always distinguish between two versions created within a minute of each other, and under these conditions the system may make an essentially random choice as to which one is preferred.

5.5 Classes of files and space allocations

Associated with each file owner is a file directory which contains information about the owner's files, an account of how much disc space has been allocated to and used by the owner, and information concerned with the status of his files. The directory itself is stored on the disc, and is not directly accessible to the user. The owner can, however, obtain details of information in the directory (see 5.11).

When a new file is created, the user may specify how it is to be stored by quoting the class of the file in the fourth component of its title. The available classes are denoted by single letters, which are defined as follows:

P Permanent file - to be held on disc and to be copied on to magnetic tape by the dump system (see below).

T Temporary file - to be held on disc only and not automatically copied on to magnetic tape.

A Archive file - to be held on disc only until it has been satisfactorily copied on to a system archive tape; thereafter not to be held on disc (see 5.9).

If the user does not quote a class letter when a file is created, then class T is assumed.

Temporary files are for information that is ordinarily required only for short periods of time, and therefore does not need to be protected against loss through system failures. Permanent files are for information required to be held for longer periods, but which must be accessible at any time; they are stored on the disc, and the fact that they are also copied on to system magnetic tapes minimizes the risk of their being lost as a result of system failures. Information that does not need to be immediately accessible may be held in archive files, and these are stored on system mag-

netic tapes only.

The class of a file may be changed by the user, and this causes the filing system to take whatever action is required to alter how the file is stored. For example, a change from class P to class A enables a permanent file to be made into an archive file; as soon as a copy has been stored on magnetic tape it is deleted from the disc. Conversely, a change from class A to class P causes an archive file to be recovered from magnetic tape to the disc. The operations of copying files to magnetic tape and recovering files from magnetic tape to the disc take place automatically, but, since magnetic tape handling must be scheduled with other work in the system, these operations cannot be instantaneous. Thus, a newly created archive file will first be stored on the disc until the system has had the opportunity to make the necessary tape copy. Further details of the archiving and file dumping mechanisms are given in 5.9 and 5.10.

The space available for users' files on the disc is limited, and it is therefore necessary to provide an allocation mechanism that controls the amounts of disc and tape used to hold the files belonging to each individual file owner. The filing system design is such that a variety of administrative procedures for file space control can be implemented with a minimum of extra system programming. Initially, a very simple system was used. Each user was given an allocation of P and T space, the difference being that class P files were dumped on to magnetic tape and class T files were not. It was planned to go to a more elaborate system in which a proportion of the T space would have been recovered by the system every morning, the object being to enable users to have more space for manipulating files during the day than they were allowed to keep permanently. However, at the time of writing, it has been decided to implement a still more elaborate system that is designed so as to encourage users to make efficient use of the disc space allocated to them; users are enabled to have relatively large amounts of space for short periods, but at the same time they are encouraged to remove files when they are no longer required.

The disc space allocated to each file owner is administered in the form of a <u>disc credit</u>, this being expressed in terms of block-days. At the start of each working day, the system automatically subtracts from each owner's credit an amount equal to the total number of blocks of disc occupied by his files at that time (this includes all class T files, and any class P and A files that are on the disc). At the same time, the owner receives a certain <u>daily income</u> which is added to his credit. Thus, for example, if a file owner receives a daily income of 20 and has files on the disc amounting to 20 blocks, his credit will remain static. If his files occupy less than 20 blocks, his credit builds up (to some limit - see below); if they occupy more than 20 blocks, his credit is reduced correspondingly.

If, at any time, an owner's credit is found to be negative, then no user is permitted to create new files or update existing files belonging to that owner; other filing operations, including reading, continue to be

available in this state. It is important to note that, if the credit becomes negative and the number of blocks of disc used is greater than the daily income, then the user must remove or delete some material from disc before he will be allowed to create or update files again. Even if this is done, file creation or updating is not permitted until his credit ceases to be negative; i.e. not until the start of the next working day at the earliest.

The initial disc credit given to a new user and his daily income are determined by the administrative staff. Associated with each file owner's credit is a credit bound (B) which is also set by the administration, and this is used to control the size of the credit in two ways:

(1) the owner's credit will not be accumulated to a value greater than B, and

(2) the credit will not be allowed to become less than -B.

Since the amount of disc space in use is counted against the credit only at the start of the working day, a user is permitted to use relatively large amounts of disc during the day without charge, so long as he deletes the files before the appropriate time. However, a user is not allowed to use unlimited space in this way, and for this reason each file owner is given a disc limit. The amount of disc space occupied by the owner's files must not be greater than the disc limit at any time.

In addition to the above mechanism for controlling the use of disc space, each file owner is given an allocation of archive space. This allocation determines an upper limit to the amount of space occupied at any time by class A files. This is an absolute allocation that is not subject to daily fluctuations; a class A file is counted against this allocation irrespective of whether it happens to be on disc or tape at any moment. The EXAMINE command can be used by the file owner to find out how much file space he is using and the current value of his disc credit, together with his credit bound, disc limit and archive allocation.

5.6 Security

The security mechanisms of the filing system enable the files belonging to an owner to be protected from accidental or malicious misuse by other users. This security relies, therefore, upon the system knowing whether the user of a file is its owner or not, so that it is necessary to prevent one user from masquerading as another. For on-line use of the filing system, the password associated with the user, which is always checked when he logs in, provides this security. An off-line job may quote a password in the job description by writing it on the same line as, and to the right of, the job title. If a password is quoted in this way, the job will only be run if the password is found to be correct. (For details, see Machine-Code Manual, Section 10.4.2). Alternatively, an off-line job can have the password checked at some stage during execution by using the 1026/7.0 extracode.

Note that a password which is 'null' (i.e. no characters) cannot be quoted in a job description. Passwords quoted in off-line jobs are potentially insecure, since paper tapes or cards submitted to the off-line service are placed in a reception area that is not guarded. To alleviate this problem, the user has a number of choices in how security is handled in off-line jobs; the choice is made when he sets his password, and further details are given in the description of the PASSWORD command in 7.2.

It has been mentioned already that the fourth component of a file title includes four letters giving the access status of the file; when a file is created, the user may specify these letters, but if he omits to do so, the letters 'FFRR' are assumed. These four letters describe the range of filing activities permitted to four different categories of users:

Owner category	the first letter specifies the activity permitted to the file owner.
Partner category	the second letter specifies the activity permitted to a partner (see below).
Key category	the third letter specifies the activity permitted to a keyholder (see below).
Public category	the fourth letter specifies the activity permitted to any user.

Partner and key categories are associated with the relationship mechanism that enables a file owner to specify the conditions under which another user may have access to his files other than as a member of the general public. The relationship mechanism is described in 5.7.

Each letter of the access status describes a particular class of activity allowed to the appropriate category of user. For example, in the access status FFRR, 'F' stands for 'free access' and 'R' stands for 'read only'. There are six different types of activity.

Load	reading the file with the intention of executing it as a program.
Read	reading the file for any purpose.
Explicit delete	deleting the file with the appropriate command or extracode.
Implicit delete	deleting the file by creating a new file with the same title.
Update	modifying the file; only applies to block files.
Change status	changing the access status or class without changing the contents of the file.

These activities are grouped together, and each group is represented by one of the letters that may be written in the access status, as follows:

F	free	all activities
U	update	all activities
D	delete	explicit delete, change status, read, load
C	change	change status, read, load
R	read	read, load
L	load	load
N	no access	no activities

The privilege of creating a new file is not included in these groups. Naturally, the file owner himself may create new files, and he may permit other users to do so if he wishes through the relationship mechanism (see 5.7). The following are some common examples of access status:

FFRR	Owner and his partners (if any) have free access; keyholders and the rest may only read or load the file.
FFNN	Owner and his partners (if any) have free access; keyholders and the rest cannot use the file at all.
FFFF	All users have free access to the file.
CRRR	Owner can change status, read or load the file only; others can only read or load it. This prevents the user from destroying a valuable file by accidentally deleting it or creating a new one with the same file title.

The filing system will not permit the access status given to the owner to be less than C, thus preventing the creation of a file that cannot be deleted by its owner.

5.7 Relationships

In order to permit another user to have special access to his files, a file owner creates a relationship which specifies the conditions that must apply before the other user may have access in a category other than public category (see 5.6). The mechanism is flexible and permits a wide range of alternatives, which are fully described below in 5.7.3. The majority of users, however, are unlikely to require all the facilities provided, and it is expected that their purpose will be served by two of the more common relationships called partners (5.7.1) and keyholders (5.7.2).

Relationships of all kinds can be created by the file owner by use of

the RELATION command; he may subsequently remove such a relationship with the EXAMINE command.

5.7.1 Partners

A file owner may create a relationship which specifies that a particular user has access to his files in partner category; the user is then known as a partner of the owner's files. A partnership may apply to all the owner's files or only to a group of such files, and, in addition, the owner may specify an upper limit to the activities that are permitted to the partner. The following examples illustrate the use of the RELATION command to create partnerships; in each case, the user (and file owner) is assumed to be AGF210.

(a) RELATION PARTNER DFH200 STATUS F

This permits the user DFH200 (who may or may not be a file owner himself) to have access to all the files belonging to AGF210 in partner category. The letter following the word STATUS is one of those described in 5.6 and indicates the maximum status permitted in partner category. In this case, the letter F denotes that the user DFH200 may have access to any file belonging to AGF210 subject only to the conditions imposed by the partner category letter specified in the access status of that file.

In addition, by virtue of the status letter F in the partnership, DFH200 may create new files belonging to AGF210. In this context, there is a difference between status letter F and status letter U; in either case, the partner is permitted otherwise unlimited access to any existing files of the owner, while status F also gives him the privilege of creating new files.

(b) RELATION PARTNER DFH200 STATUS R

In this example, the user DFH200 is permitted to access the files belonging to AGF210 with a maximum status R in partner category. This means that he would be permitted only to read or load a file whose access status is, for example, FRNN or FFNN. He is not permitted to access a file with status FNNN, but he would have free access to a file with status FFFF, since he still ranks as a member of the general public.

(c) RELATION PARTNER DFH200 GROUP XYZ

This example shows how a user may be made a partner for a particular group only of the owner's files, a group being defined to be all files with a certain second component in their titles. In this case, the partnership applies only to files with titles of the form AGF210/XYZ/... When the status has been omitted, as in this example, status letter F is assumed.

5.7.2 Keyholders

The keyholder form of relationship enables special access to an owner's files by any user who can quote a certain keyword either explicitly or by program. The mechanism is similar to that for partners (5.7.1): the file owner gives the keyword to be quoted (rather than a user's identifier), and a status letter which determines the maximum activity that may be used in key category. The relationship can be made to apply to a group of files.

The following examples illustrate the keyholder mechanism; again, it is assumed that the user(and file owner) is AGF210.

(a) RELATION KEY STATUS F

Whenever a keyword is involved, the command responds with a line of 8 superimposed characters over which the user types the characters of the keyword itself. In this example, any user who can subsequently quote this keyword has free access (including the ability to create new files) to any of AGF210's files, subject to the key category status set for any existing file.

(b) RELATION KEY STATUS R

(c) RELATION KEY GROUP XYZ

These two examples are similar to the examples (b) and (c) given in 5.7.1.

Any user who wishes to make use of a keyholder relationship so as to gain access to the files of another owner in key category, must first prove to the system that he knows the keyword. This can be done by using the UNLOCK command. The command responds with the usual line of 8 superimposed characters, over which the user types the keyword. Subsequently, one or more accesses may be made to the other owner's files, until such time as a further keyword is quoted to the system; this may be done explicitly by use of the UNLOCK command, or implicitly by a program or sub-system called by the user, using the 1026/11.0 extracode.

5.7.3 General relationships

This section gives complete details of the relationship mechanism, of which partners and keyholders are special cases. The average user does not need to be conversant with these details.

Access status and categories

The four status letters - one for each category - associated with individual files have already been described. At the moment that a user attempts some activity concerned with his own or another's files, a further

group of four status letters - again one for each category - are derived by the system in a manner explained in detail below. These are known as user status letters, and they indicate the degree of privilege that the user has in relation to the files with respect to which he is attempting the activity.

User status

In determining the status of the user at the moment of attempting some activity, the filing system checks a set of conditions which are associated with the file owner's directory. The conditions include a set of standard conditions (which apply to all file owners) and the set of relationships which have been set up by the file owner. Each condition can be expressed as some test (or tests) which, if satisfied, specifies a set of four status letters which apply at that moment. Thus, there may be several such sets of status letters, and the user status is determined by combining these sets taking the 'highest' status in each of the four categories. For example, if the tests produce the sets NRNN and NNNU, then the user status is NRNU; that is, there is permission to read or load a file which has at least that amount of status in partner category, or to do anything with a file which has at least that amount of status in public category.

Standard conditions

The standard conditions, which apply to all file owners, are as follows:

(a) If the user is the file owner and is on line, then status FNNN applies.

(b) If the user is the file owner, if he is operating off line and has satisfied the conditions with regard to his password being checked (see 5.6), then status FNNN applies.

(c) Status NNNU applies, whether or not the user is the file owner.

Activity rules

The rules followed by the filing system in checking whether any particular file activity is permitted or not, are as follows:

(1) If the activity is connected with an existing file, the activity is allowed if, in any one of the four categories, the letters of the file status and the user status both specify sufficient authority.

(2) If the activity is to create a new file, or to examine the directory entry of an existing file, a relationship or space allocation, then the letters of the user status must specify status F in at least one of the categories: owner, partner, key. Note that the user must rank as more than a member of the public, whatever status has been

associated with that category.

(3) If the activity is to create or delete a relationship (for example, a partner or keyholder), then the user status must have status F in owner category.

Relationships

Relationships may be created (and subsequently removed) either by the file owner himself, or by any user for whom a relationship has already been created nominating him as permitted to operate in owner category with status F. Relationships are implemented through special entries in the file owner's directory; they may be created by the RELATION command (or the 1026/10.0 extracode) and removed by the EXAMINE command (or the 1026/10.4 extracode).

A relationship consists of up to 4 tests together with a set of 4 status letters; if all the tests are satisfied, then the status letters will be used in evaluating the user status. Each test may be any one of the following types, with the restriction that not more than 2 of them may be the kind that have parameters (that is, the first 4 types).

If user identifier is x
If user has quoted key x
If user is referring to a file whose second component is x
If user is obeying the system command x
If user's password has been checked
If user is on line.

The following examples illustrate the use of the RELATION command; in each case a relationship is created for the current user's files, unless the SET command (see 5.2) has been used previously.

(a) RELATION USER DFH200 PASSWORD GROUP XYZ STATUS NFNN

If the identifier of the user is DFH200, if his password has been checked, and if he is referring to those files of the owner with a second component XYZ, then user status NFNN is permitted; in other words, if DFH200 has had his password checked he can do anything in partner category to files /XYZ/..., including creating them.

(b) RELATION COMMAND ABC ONLINE STATUS NNUN

If the user is obeying the command ABC and is logged in at a console, then access to the owner's files is permitted with user status NNUN; note that any user may thus be permitted this type of access, but only via the stated command. This kind of relation can be employed to permit a system command to access system files irres-

pective of which user has called the command.

(c) RELATION KEY STATUS NRFN

In this case, the command responds with the usual row of 8 superimposed characters, over which the user types the keyword. Any user who can quote this keyword (using the UNLOCK command or 1026/11.0 extracode) has user status R in partner category and user status F in key category.

5.8 Interlocks

To avoid the unpredictable effects that may arise when two users attempt to operate simultaneously on the same set of information, the system provides the following interlocks in the use of files.

While a file is open for reading or loading:

(1) The file may be further opened for reading or loading by the same or another user.

(2) The file may not be further opened for updating, and it may not be deleted; also a new file may not be created with the same title.

While a file is open for updating:

(1) The file may not be further opened for any purpose, and it may not be deleted; also a new file may not be created with the same title.

An attempt to use a file which violates these rules gives rise to a filing error (see 5.12). Note that the operations of changing the access status or class of a file and setting a preferred version are not subject to the interlock rules, and may be done at any time.

5.9 Archive mechanism

As described in 5.5, class A files are ordinarily stored on archive tapes and not on the disc. This mechanism provides the means by which a user may store away infrequently required material, and have it recovered to the disc at some future time. The user operates the mechanism simply by setting or changing the status of a file as required; thereafter, the system automatically arranges for the necessary disc and magnetic tape transfers to be performed. Archive tapes are also used by the system to store copies of the owner's class P files as part of the mechanism for guarding against loss of information in the event of disc failures (see 5.10). In fact, the only essential difference in the treatment of class P and class A files is that class A files are deleted from the disc when they have been copied to archive tapes, whereas class P files are not.

The system operates as follows. At intervals of about one week, archive tapes are updated to contain a copy of every class P and class A file. This action is entirely automatic and outside the direct control of the user, except that he can speed up the process by use of the ARCHIVE command. Once a file has been protected in this way, it will be held on the system tapes until about one week after it is made obsolete by user action (for example, if a new file is created with the same title or if the file is deleted).

Whenever a file (class A or P) is copied to a system tape, the system assigns to the file an archive name. (This name is unique only to different versions of the same file, and different files may be given the same archive name.) The archive name is given as part of the information produced when a directory is inspected by the EXAMINE command. To the user, the only purpose of this name is connected with the recovery of files that have become corrupted or have been deleted accidentally. (See 5.10.)

A class A file is normally recovered from the archive tapes to the disc either by changing it to class P or by using the RECOVER command. The system then automatically arranges for the recovery to take place, and usually this takes about an hour or so. Note that owing to the way in which the mechanism operates, it is not permitted to recover an archive file by changing it to class T. For similar reasons, a class P file may not be changed to class T if, at the time, the file is not on the disc (i.e. it is in the process of being recovered from magnetic tape). Further details of the recovery mechanism are given in the next section.

5.10 File dumping and recovery

In addition to being copied to archive tapes at weekly intervals, class P and A files are also automatically copied to system dump tapes more frequently. At various times during the day (currently 3 hourly intervals) the filing system initiates a general dumping procedure, which writes to a dump tape a copy of any class P or A file that has been created or updated since the last dumping operation took place. The purpose of the dumping procedure is to save copies of files on magnetic tape so that if a file is lost for some reason, the system can recover either an up-to-date version or at least a version that is not very old. Dump tapes provide the normal back-up protection for recently created or updated class P or A files. On the other hand, versions of files that have not been updated for some time are held on archive tapes. This dual mechanism both provides comprehensive back-up facilities in the event of file loss, and enables efficient use to be made of magnetic tapes.

The recovery mechanism provided for retrieving archive files is also used to restore copies of class P files that have been lost or corrupted in some way. Whenever such a file is lost as a result of a hardware or system failure and the fact is detected by the system, the recovery mechanism is activated automatically. The most up-to-date version of the file that

can be found is retrieved from a dump tape or an archive tape whichever is appropriate.

It is always possible that a file on the disc may become corrupted without the knowledge of the filing system. Under these circumstances, only the user can detect that this has occurred, and in this event the file owner can invoke the recovery operation by using the RECOVER command (or the 1026/6.0 extracode). For example:

```
RECOVER AGF210/PROG/A1
```

This facility operates by causing the filing system to throw away the disc copy of the relevant file, so that the recovery mechanism is activated to retrieve the latest version of this file that can be found on a system tape. As a result of this, it is important to note that if there is no copy available on tape (that is, if the file is class T, or if a file with this title had not been dumped previously), then the only effect is to delete the file. A user may alternatively request that a particular version of a file which has previously been copied to an archive tape is to be retrieved. In this case, he quotes the archive name of this version when the RECOVER command is used.

Similar use may be made of the RECOVER command to retrieve copies of class P or class A files that have been deleted accidentally. Note, however, that this can only be done if a version of the file exists on a system tape at the time. After a file has been deleted, the archive and dump mechanisms retain copies of this file only for so long as is convenient for efficient magnetic tape usage; in many cases, however, this is long enough for recovery to be possible.

5.11 Directory inspection

The directory associated with each file owner contains all information necessary to enable the system to handle each of the owner's files. The directory also contains the information concerning file space allocations, together with the relationships that the owner may have created to enable other users to have special access to his files. This information is the property of the filing system and does not belong to the file owner, although examination of it is permitted to the file owner and to any user who, by virtue of a relationship, is permitted to create the thing about which he is asking for information.

Information concerning one or more of an owner's files can be obtained by using the EXAMINE command or the 1026/3.0 extracode. Information concerning the owner's file space allocations and relationships can also be obtained by the EXAMINE command.

The precise details of any part of the directory are similarly available through the 1026/4.0 extracode (there is no equivalent command at present). This enables the directory to be interrogated 'down to the last

bit', and details of how this information is interpreted are given in Appendix 4.

5.12 File errors

There are a number of error conditions that can arise whenever some file activity is attempted by a user. In all cases, these are reported in the form of a basic system fault, where the fault number is 20 and the value of b91 indicates the particular nature of the error (see Machine-Code Manual, Chapter 9). This is known as filing fault n, where n is the value of b91. When one of these errors occurs, it is usual for the command or subsystem being used to interpret the filing fault number in terms of a suitable message.

A complete list of filing fault numbers with descriptions of their meanings is given in Appendix 2.

CHAPTER 6 - PRIVATE COMMANDS

The system is organised so as to make it particularly easy for a user to create his own private commands. A private command is created by filing away the appropriate program, and in fact the only major difference between a private command and a system command lies in the directory in which the name of file appears (see 7.1). The following sections describe the techniques and some of the conventions for constructing commands.

6.1 Command filing

A command is constructed by creating a block file that contains the absolute binary form of the appropriate program. This program must be written so that it is correctly entered when it is called by the command system as described in 6.2. There are a number of techniques available; which one is chosen in any particular case depends among other things on the total amount of storage space required for program and data, and on how much of this is occupied by the program itself.

(a) Any program size and any store size. The system dump mechanism (see 4.5) is the simplest but not necessarily the most efficient technique. This mechanism enables a program of any size to be dumped in a file in such a manner that it can be called directly by the command system, but since the process of calling the program entails two successive requests to the operating system for core space, this technique is not as efficient as some of the alternatives described below.

(b) Program size ⩽ 512 words, total store space used ⩽ 4096 words. The program should be written so as to reside in the first 512 words of its storage space, and with the entry point at address 0. The program in binary form is then simply copied to block 1 of the file.

(c) Program size ⩽ 4096 words, total store space used ⩽ 4096 words. The program should be written so as to reside in the lower addressed part of its storage space, and with the entry point at address 0. The program in binary form should be copied to consecutive blocks of the file starting at block 1. Since the command system will only load block 1 of the program when it is called, the program must be written so that when it is entered it loads the remaining blocks. Note that on entry to the program the file will be open for reading as BSD 126 and positioned at block 2.

(d) Store size > 4096. In this situation it is normal to use (a) above. If this is inappropriate for some reason then method (c) above may be used, provided that the command is always called with sufficient space specified in the environment declaration. Alternatively, the following

technique may be adopted. A suitable bootstrap program is written whose purpose is to request the supervisor to allocate the appropriate amount of core space and then to load the program of the command itself (both these actions are provided by the 1143 extracode). Assuming that the bootstrap program is itself not more than 512 words long, it can be filed in a single block; the program of the command proper is conveniently held in subsequent blocks of the file, although the user is free to make other arrangements if he so desires.

6.2 Entry and exit

The command system calls a private command by using the 1146 change phase extracode which has the following effect. The appropriate file is opened, for reading only, as BSD 126, 4096 words of store are requested (unless otherwise specified in the environment declaration), block 1 of the file is loaded into the first block of the store, and control is transferred to address 0. At the time of calling, parameters may be handed from the command system to the program as described below in 6.3.

The ordinary way for a command to terminate is by executing the 1117 end program extracode, or its equivalent in other languages. This facility has a dual purpose. If the program is executed in a normal off-line job, the extracode causes the job to be terminated; on the other hand, if the program is called by the command system (on line or off line) control is returned to the command system for the next command. These rules also apply if the program terminates through the system monitor mechanism.

When control returns to it, the command system detects whether or not the command failed. In this context, a failure is indicated in one of the following circumstances:

(a) the program terminates through the system monitor mechanism,

(b) the program terminates in a private monitor sequence,

(c) the program terminates via an 1117 extracode specifying a negative address.

If the command which had called the program was part of a COMMAND command sequence (see 4.6) or part of an off-line command sequence (see 4.7), then that command sequence is abandoned. Also, in case (c), if the address in the 1117 extracode is of the form n.1J4, then the command system generates a suitable error message for system fault n on behalf of the program. This facility saves the writer of the command from the need to provide extensive message reporting in his own program. (If the system fault is of the kind which has further information specified in b91, then the name of a b-register holding this value should be written as the Ba part of the 1117 extracode.)

6.3 Parameter handling

A private command may be written so as to perform its task according to parameters specified at the time of call. There are a number of independent techniques available, ranging from conventional handling of input and output streams to the specification of parametric information in almost any desired format.

(a) Argument string

As described in 4.2.3, an argument string may be specified when any command is called, and with a few restrictions this string may contain an arbitrary sequence of characters. The argument string is particularly useful for specifying file titles and other simple directives; it is especially appropriate when a command is being written with a view to wide public use and it is important that the command should be easy to use.

The argument string is always handed to the program of the command in two ways. First, all the characters of the string are available on input stream 253; if there is no argument string, then this stream will not exist. Secondly, the argument string is processed by the command system on the assumption that it contains items in a certain standard format, and the results of this processing are stored in successive b-registers from b1 to b69. This format includes file titles (in the conventional form), words (i.e. strings of letters), integers (strings of digits), and other characters. SPACE and TAB characters may separate two items where necessary to avoid ambiguity. The successive items recognised are each indicated by a sequence of b-registers, the first of which gives the type of the item and the remainder, according to the type, give the information itself as indicated in the following table.

Type	Description	How stored
0	end of data	-
0.4	file title	left justified in next 7 b-registers, filled out with 0.5 characters
1	word	left justified in next 2 b-registers, filled out with 0.5 characters
1.4	integer	justified to d3 in next b-register
2	other character	d0-5 of next b-register, d6=1 if character is in outer set
-ve	decoding failed	(more items than b-registers available, or the argument string is more than 1 line in length)

In addition to the above, the type of the item will have:

(i) 0.1 added, if the item is apparently faulty (that is, an integer overflowed, a word is more than 8 letters,

or a file title is more than 28 characters or incorrect in some way);

(ii) 0.2 added, if the item is not separated from the next by a SPACE or TAB character.

(b) Current document and results document

If the main function of a command is to process data from a single input stream, then it is conventional to use input stream 1; usually it is either some file which has already been opened before the command is entered (for example, by an environment declaration), or it is an unfiled (temporary) document which is the by-product of some previous command.

There are two conventions for command output, either or both of which may be used. Output stream 0 is conventionally used for diagnostics and simple results which are ordinarily to be displayed directly on the console. On return to the command system from a non-interactive command, this output stream is usually renamed as input stream 252 (the results document), and a copy of it is then sent to the console. If the information produced by the command is to form the current document the following technique may be used. The command produces the information on some convenient output stream (which it creates for this purpose); then before the command terminates, the output is renamed as input stream 1, thus producing a new current document.

(c) Other input/output streams and backing-store devices

At the originator's discretion a private command may also use other streams and BSD's both for external communication and for internal working. It is conventional to use low numbered streams and BSD's for external communication, and high numbered streams and BSD's for internal working. Note that it is good practice to close at least all the internally used streams and BSD's before terminating the command.

(d) Interactively obeyed commands

An interactively obeyed command is permitted to communicate directly with the console. Conventionally this is done through input stream 0 and output stream 0 which are connected to the console by the command system before the command is entered (see 4.3). Once entered, the command may disconnect these streams and connect other streams to the console or to any other logical consoles associated with the user (see 1.6).

6.4 Security and errors

There are two aspects of file security that may be relevant when a private command is intended to be used by someone other than the owner of the files concerned. First, there is the security of the file containing the program itself; absolute security is guaranteed if the relevant users have access to the file with status L (load only). Secondly there is a

problem concerned with other files (belonging to the owner of the command) which are used and possibly updated by the program. In this case it is necessary to arrange that the program has the necessary privilege to access such files regardless of which user calls the command. A useful technique is to employ a keyholder relationship (see 5.7.2) and to arrange for the program to quote the key before opening the file; for further security, the program should arrange both to close any such file and to remove the keyholder privilege by quoting a null key before terminating. These actions must be taken even if the command is unexpectedly stopped as the result of an error or by a quit signal from the user; to do this, the private monitor mechanism must be used.

If a user is writing a private command which is to have widespread application - especially if it is destined to become a system command - it is desirable to have some regard to the ease with which it can be used, the format of its parameters (if any), the way in which it communicates results and, most important, the way in which it deals with errors when they occur. The writer should arrange for suitable actions to be taken when errors and faults arise directly as a consequence of what the program is attempting to do. He should also be prepared to deal with faults that may arise at unpredictable times, for example, if the user types @Q, @I or @M at any time while the program is being obeyed.

CHAPTER 7 - COMMAND SUMMARY

7.1 Categories of commands

Commands can be grouped into the following categories:

(a) system commands,
(b) auxiliary system commands,
(c) private commands.

With the exception of certain system commands (which are stored as part of the command program for reasons of internal efficiency), each command is held as a file containing the appropriate program in binary format. System commands are always called by typing their command names. Auxiliary system commands are held in files owned by L, and are called by typing the corresponding file titles; for example, L/WHO. Private commands are held in files belonging to private libraries or to individual file owners, and are called by typing the appropriate file titles.

New commands may be added to the system at any time, and specifications of all system and auxiliary system commands are given in the separate document entitled CAMBRIDGE MULTIPLE-ACCESS SYSTEM, COMMAND SPECIFICATIONS. The next section contains summary descriptions of the most important system commands. Appendix 6 contains a table of these commands for reference purposes.

7.2 Summaries of important commands

For each command described, one or more formats are given indicating the parameters that may be specified. This is followed by a brief description of the actions performed by the command. The descriptions are mainly intended to give the reader sufficient information on how the commands are normally used; users of the system may need to refer to the COMMAND SPECIFICATIONS document for further details.

Certain commands operate directly on input and output streams whose numbers are specified as parameters. In these cases, standard formats are used which closely resemble those of environment declarations, and these formats are defined in groups as follows:

(a) 'input stream' stands for one of the forms

Pn Dn	input stream n
Z	the results document (input stream 252)
H	a HERE document as in an environment declaration (see 4.2.2).

(b) 'output stream' stands for the form

On output stream number n

(c) 'stream' stands for either an input stream or an output stream.

(d) 'document' stands for a stream or a file title. In many cases where a parameter is specified to be a document, the absence of any parameter written indicates the current document.

In each command format given below, upper-case letters are used for the command names while parameters are described by one or more letters or words in lower case.

ARCHIVE file owner

Files are automatically copied to archive tapes periodically, and this command enables a user to request an early copying of his, or some other owner's files in order to clear unwanted class A files from the disc.

AUTOCODE documents

Calls the Titan Autocode system to compile and run a source program given as the current document or in the specified documents. In the latter case, these are opened, or renamed as appropriate, as input streams 1, 2, ... (P-type).

CLOSE
CLOSE streams

Closes the specified streams; if no parameters are specified, closes all streams and backing-store devices, with the exception of any specified in an environment declaration given with the command.

COMMAND document parameter list

Executes the sequence of commands stored in the current document or the specified document, with parameter substitutions if necessary (for details see 4.6).

```
COPY  file title  file title                  (a)
COPY  file title  file title  k               (b)
COPY  file title  file title  i  j  k         (c)
```

Copies all or part of the first file to the second, according to the parameters as follows:

(a) the entire file is copied;
(b) the first k blocks of the file are copied;
(c) k blocks are copied from block i of the first file to block j of the second file.

(b) and (c) are relevant only to block files.

CURRENT document

Opens the specified file, or renames the specified stream, as the current document.

DELETE documents

Deletes the specified files and/or streams. Note that the operation of deleting an input stream is the same as closing it.

EDIT documents

Calls the editor which operates on the current document; if documents are specified as parameters, these are opened or renamed as input streams 1, 2, ... etc. The editor first responds with

EDITOR READY

and after this the user may type editing requests as defined below. At any time there is a current line which is being pointed at; initially, the pointer is positioned before the first line of the document. In the following definitions of editing requests, Ø stands for some character chosen by the user to delimit character strings that he types; this may be any character (including SPACE) which does not occur in the string or strings that it delimits. In addition, the final delimiter of any request may be replaced by a RETURN. One or more requests may be typed on a single line; after executing each such line, the editor responds with a copy of the current line preceded by its line number.

Requests that establish a line as the current line

Mn	move pointer to the nth line of the document.
FØsØ	find the next line beginning with the string s.
LØsØ	locate the next line containing the string s.
N	move pointer to the next line of the document.

Requests that change the current line

AØsØtØ	after the string s insert the string t.
BØsØtØ	before the string s insert the string t.
EØsØtØ	extract the string s and replace it with the string t.

Editing of the current line begins at a character pointer

(not to be confused with the line pointer defined above) which is initially positioned at the beginning of the line. The following directives are associated with this pointer:

↑	move the character pointer one position to the right.
#	erase the first character to the right of the pointer.
←	move the character pointer back to the start of the line (this operation occurs automatically when a new current line is selected).

Requests that operate on whole lines

In	insert, before the nth line, the line or lines of information which follow; a Z character on a line by itself terminates this information.
Rn m	replace lines n to m inclusive with the line or lines of information which follow; a Z character on a line by itself terminates this information. If m = n, then m may be omitted.
Dn m	delete lines n to m inclusive; if m = n then m may be omitted.

In any of the above, a line number may be replaced by

.	indicating the current line.
*	indicating the end of the document.

Other requests

GØsØtØ	replace the string s by the string t wherever it appears in the current line and in all succeeding lines of the document.
HnIm	insert the document on input stream m before line n.
*	position the line pointer to the head of the document.
W	finish editing, and return to the command system with the amended document as the current document.

The editor provides a number of other features, and these are described in the complete specification of EDIT.

ETAL
ETAL file title

Calls the ETAL symbolic assembler to assemble, load, and run

a source program given as the current document or in the specified file.

EXAMINE directives

Permits the user to inspect his own or, under certain circumstances, another user's file directory. According to the directives specified, information can be requested concerning files, relationships, and allocations of file space; the directives and their actions are as follows: (the first word of any directive may be abbreviated to its initial letter if desired).

FILES	to request information about files; for example titles, dates of creation, etc.
RELATIONS	to request information about relationships.
SPACE	to request details of file space allocated and used etc.
OF file owner	specifies which file directory is to be inspected (in default the user's own directory is taken).
TITLES	to request that only the titles of files be given.
CHECK	to request interactive operation (see below).
LIST	to request information without interaction (in the absence of either CHECK or LIST, CHECK is understood).
PRINT	to send information to the printer rather than to the console.

In CHECK mode, EXAMINE pauses after typing each file specification or relationship. The user may either proceed to the next item by simply typing a RETURN, or he may type one of the following:

DELETE		to delete a file or relationship.
STATUS	status value	to change the access status or class of a file.
PREFER		to prefer a version of a file (see 5.4).
RECOVER		to initiate recovery of a file (see 5.10).
LIST		to continue without interaction.
QUIT		to abandon the command and return to command status.

E3

Calls the non-interactive line editor provided by Titan Library Routine E3.

FIGARO

FIGARO is an interactive JOSS-like system which enables small numerical calculations to be performed directly from the console.

FILE file title

Creates a new file with the given title from the current document; the file title may include a status and class if necessary.

FINISH

Logs out the user at the end of a console session (see 3.3).

F4
F4 file title

Calls the ASA Fortran system to compile, load, and run a source program given as the current document or in the specified file.

HELP words

Attempts to give relevant information on various aspects of the system as described by the words typed. In particular, HELP is useful for finding out about system commands, fault numbers, and extracode specifications.

IIT documents

Calls the IIT symbolic assembler to assemble and run a source program given as the current document or in the specified documents. In the latter case these are opened or renamed as appropriate, as input streams 1, 2, ... (P-type).

INPUT

Enables a new document to be typed. The system first responds with

*** READY

after which the user types the information and eventually terminates it by @Q. The material then becomes the current document.

LISP

Calls the LISP interpretive system to execute a source

program given as the current document.

LIST
LIST documents

Arranges for a copy of the current document, or copies of the specified documents, to be printed with line numbers on a line printer in the computer room. In the case of an input stream, the document is printed in its entirety whatever the state of the stream; an output stream is first renamed as the results document.

MESSAGE
MESSAGE name

Obtains a personal or public message sent by the SIGNAL command. The first form, when no name is mentioned, causes any personal messages sent to the user to be typed. A public message is obtained by including a name as a parameter; the following names are permitted:

DAY	gives the message of the day.
MORE	gives the 'MORE' message, which is sometimes associated with the message of the day.

MLS

Enters the Titan Mixed Language System to execute MLS commands given as the current document. (MLS is used to control FORTRAN and assembly language compilations, including, if necessary, the relocation and loading of compiled programs with the standard linking loader (IAL); a number of other language compilers are available through MLS, and the majority of these produce code which can also be loaded by IAL.)

PASSWORD digit

Used to change the user's password. The new password is typed after the command has responded with 8 superimposed characters. The digit indicates the required mode of the password (if omitted, mode 0 is assumed), and this determines under what conditions the filing system will recognise the user identifier quoted in the job description of an off-line job.

0	user identifier is recognised whether or not the password is given.
1	user identifier is recognised only if the password is given.
2	user identifier is recognised, but the password must not be given.

3 user identifier is never recognised off line, and the password must not be given.

PATCH
PATCH file title

Enters the interactive PATCH system to enable a core image to be interrogated and amended if necessary. The command operates on the core image stored as backing-store device 0; if a file title is typed this file is opened as BSD 0; otherwise, if BSD 0 does not exist, the file /DUMP is opened as BSD 0. The command first responds with

PATCH READY

and after this the user may type requests, some examples of which are given below; full details are given in the complete specification of this command.

D	indicates that a dump is being interrogated (PATCH may also be used to operate on other forms of core image).
Tp q	types the contents of address q in style p (the styles are: 0 instruction format, 1 address format, 2 octal, 3 floating point, 4 decimal, 5 character format)
Tp Bn	types the contents of b-register n in style p.
+	(after a T request) types the next location or b-register.
F d	finds the address of the first location that contains the value d (a number of formats of d are permitted).
S q d	sets the contents of address q to be the value d (a number of formats of d are permitted).
S Bn d	sets the contents of b-register n to be the value d.
+ d	(after an S request) sets the next location or b-register.
J q	loads the program and restarts it at address q (this also causes exit from PATCH).
W	closes the file and exits from PATCH.

PDP

Organises the transfer of information held in the form of a stream to and from the PDP7 (or PDP9) satellite computer. The user of this command must be logged in at the console situated with the satellite computer.

PREFER file title

The specified file becomes the preferred version of all files belonging to the particular file owner and with the particular second component (see 5.4).

PRINT
PRINT documents

Arranges for a copy of the current document, or copies of the specified documents, to be printed on a line printer in the computer room. In the case of an input stream, the document is printed in its entirety whatever the state of the stream; an output stream is first renamed as the results document.

PROJECT n

Types information about the allocations of time and the controls associated with project number n. A user is only permitted information concerned with project numbers to which he has access.

RECOVER file title
RECOVER file title archive name

Requests the specified file to be recovered from a system magnetic tape. This command may be used to recover a file that has been deleted accidentally, or that has become corrupted. In normal use, the most recent version of the file that has been dumped on a system tape is recovered; alternatively, an archive name can be specified in order to obtain a particular version from the archive mechanism (see 5.9).

RELATION directives

Enables any form of relationship for the user's files to be created (removal of such a relationship is possible with the EXAMINE command). One or more of the following directives may be used to specify the conditions of the relationship (see 5.7.3).

USER user identifier
COMMAND command name
GROUP file title component
KEY
ONLINE
PASSWORD

If KEY is used, the user types the keyword after the command has responded with 8 superimposed characters.

The directive

STATUS abcd (where 'abcd' are 4 status letters)

may also be included, and this specifies the status letters that the relationship is to provide in each of the four categories. If only one status letter is given, then this status applies to partner category only, unless KEY is used, in which case the status applies to key category only. If no STATUS directive is given, then STATUS F is assumed. All directives may be abbreviated to their initial letters.

RUNJOB document parameter list

Initiates an off-line job whose job description is given as the current document or in the specified document. The job description must be in the standard format used for off-line jobs. A user may create several off-line jobs during an on-line session, provided that only one is awaiting execution or in the process of execution at any time. A parameter list may be given, in which case this has the same format and is dealt with in the same manner as for the parameters of a COMMAND command (see 4.6.1).

SAVE input stream

The current document is renamed as the specified input stream.

SET file owner

Sets the 'understood file owner' to be that specified. This means that any subsequent file titles which have the first component omitted are taken as belonging to this file owner, rather than to the user. This command also affects the actions of the ARCHIVE, EXAMINE, FSPACE, PASSWORD, and RELATION commands.

SETDUMP

Sets the dump switch to indicate that a program subsequently called by the user is to be dumped in a file with the title /DUMP//FFRR.T if that program terminates through the system monitor mechanism or executes an 1116 extracode (see 4.5). The dump switch may be unset by using the same command, and specifying the word UNSET as a parameter.

SIGNAL user identifier (message)

Sends a personal message to the specified user. The message may contain several lines of information with the only restriction that any brackets therein must be correctly paired. If no message is specified, the information is taken from

the current document. The message is held by the system in a special file until it is requested by the recipient using the MESSAGE command; the recipient is signalled with '*** MESSAGE FOR YOU' when he is next in command status if he is logged in, or when he next logs in.

STATUS file title

Changes the access status and/or class of the specified file as given by the fourth component of the title.

STREAMS

Types a summary of all currently existing input and output streams and backing-store devices, giving their numbers and some details of how they have been created.

TIME

Types how much time the user has been allocated and has used in the current on-line session. Both the COMP and EXEC times are given, and these are accurate to about 5 seconds.

TYPE documents

Types a copy of the current document, or copies of the specified documents on the user's console. In the case of an input stream, typing begins at whatever point the stream is positioned at in the document; an output stream is first renamed as the results document.

UNLOCK

Enables the user to quote a key in order to get access to some other file owner's files. The key is typed after the command has responded with a line of 8 superimposed characters.

APPENDIX 1 - TELETYPE/INTERNAL CODE TABLES

Three tables are given corresponding to internal code inner set, outer set and carriage control set. In each table the column headed INPUT indicates how each character can be input from a Teletype, and the column headed OUTPUT indicates the effect on characters being output to the Teletype. The effect of typing any Teletype character not mentioned is treated as an error, and gives the response *** FAULT CHARACTER; and does not enter any input.

Notes for the tables

1. When TAB is typed, a single space appears on the printed page.
2. When LINE FEED is typed, the system responds with a RETURN.
3. When RETURN is typed, the system responds with a LINE FEED.

INNER SET

CODE & CHARACTER	INPUT	OUTPUT	CODE & CHARACTER	INPUT	OUTPUT
0.0		.	4.0 '	'	'
0.1 space	space	space	4.1 A	A	A
0.2 tab	TAB[1]	space	4.2 B	B	B
0.3 backspace	@B	.	4.3 C	C	C
0.4 shift OUTER		no effect	4.4 D	D	D
0.5 shift INNER		no effect	4.5 E	E	E
0.6 LC/LS	CTRL Y	no effect	4.6 F	F	F
0.7 UC/FS	CTRL U	no effect	4.7 G	G	G
1.0 (	(	(	5.0 H	H	H
1.1)	)	)	5.1 I	I	I
1.2 ,	,	,	5.2 J	J	J
1.3 π	@H	.	5.3 K	K	K
1.4 ?	?	?	5.4 L	L	L
1.5 &	&	&	5.5 M	M	M
1.6 *	*	*	5.6 N	N	N
1.7 /	/	/	5.7 O	O	O
2.0 0	0	0	6.0 P	P	P
2.1 1	1	1	6.1 Q	Q	Q
2.2 2	2	2	6.2 R	R	R
2.3 3	3	3	6.3 S	S	S
2.4 4	4	4	6.4 T	T	T
2.5 5	5	5	6.5 U	U	U
2.6 6	6	6	6.6 V	V	V
2.7 7	7	7	6.7 W	W	W
3.0 8	8	8	7.0 X	X	X
3.1 9	9	9	7.1 Y	Y	Y
3.2 <	<	<	7.2 Z	Z	Z
3.3 >	>	>	7.3 runout	@←	no effect
3.4 =	=	=	7.4		.
3.5 +	+	+	7.5		.
3.6 -	-	-	7.6		.
3.7 .	.	.	7.7 fault	@%	#

OUTER SET

CODE & CHARACTER	INPUT	OUTPUT	CODE & CHARACTER	INPUT	OUTPUT
0.0		.	4.0 suffix 10	@*	.
0.1 space	space	space	4.1 a		A
0.2		.	4.2 b		B
0.3		.	4.3 c		C
0.4 shift OUTER		no effect	4.4 d		D
0.5 shift INNER		no effect	4.5 e		E
0.6 LC/LS		no effect	4.6 f		F
0.7 UC/FS		no effect	4.7 g		G
1.0 ∧	@&	&	5.0 h		H
1.1 ∨	@+	V	5.1 i		I
1.2 ×	@X	.	5.2 j		J
1.3 ;	;	;	5.3 k		K
1.4 stopcode	@.	no effect	5.4 l		L
1.5 $	$	$	5.5 m		M
1.6 bell	bell	bell	5.6 n		N
1.7 :	:	:	5.7 o		O
2.0 ∅	@$	.	6.0 p		P
2.1 [	[	[	6.1 q		Q
2.2]	]	]	6.2 r		R
2.3 →	@A	.	6.3 s		S
2.4 ≥	@>	.	6.4 t		T
2.5 ≠	@=	.	6.5 u		U
2.6 underline	@U	.	6.6 v		V
2.7 |	@V	.	6.7 w		W
3.0 %	%	%	7.0 x		X
3.1 ≈	@S	.	7.1 y		Y
3.2 @	@@	@	7.2 z		Z
3.3 ←	←	←	7.3 ~	@T	.
3.4 ↑	↑	↑	7.4 §	@P	.
3.5 \	\	\	7.5 #	#	#
3.6 "	"	"	7.6 !	!	!
3.7 suffix 2	@,	.	7.7 erase	@#	#

CARRIAGE CONTROL SET

CODE & CHARACTER	INPUT	OUTPUT	CODE & CHARACTER	INPUT	OUTPUT
0.0 null	@Q, @I or @M	no effect	4.0 PAGE	FORM FEED	RETURN + 5 LINEFEEDS
0.1 1LF	LINEFEED[2]	1 LINEFEED	4.1	VT	"
0.2 2LF		2 LINEFEEDS	4.2		"
0.3 3LF		3 "	4.3		"
0.4 4LF		4 "	4.4		"
0.5 5LF		5 "	4.5		"
0.6 6LF		6 "	4.6		"
0.7 7LF		7 "	4.7		"
1.0 8LF		8 "	5.0	@0	"
1.1 9LF		9 "	5.1	@1	"
1.2 10LF		10 "	5.2	@2	"
1.3 11LF		11 "	5.3	@3	"
1.4 12LF		12 "	5.4	@4	"
1.5 13LF		13 "	5.5	@5	"
1.6 14LF		14 "	5.6	@6	"
1.7 15LF		15 "	5.7	@7	"
2.0 CR		RETURN	6.0	ALT MODE	"
2.1 CR + 1LF	RETURN[3]	RETURN + LINEFEED	6.1	EOT	"
2.2 CR + 2LF		" 2 "	6.2	RUBOUT	"
2.3 CR + 3LF		" 3 "	6.3	WRU	"
2.4 CR + 4LF		" 4 "	6.4		"
2.5 CR + 5LF		" 5 "	6.5		"
2.6 CR + 6LF		" 6 "	6.6		"
2.7 CR + 7LF		" 7 "	6.7		"
3.0 CR + 8LF		" 8 "	7.0		"
3.1 CR + 9LF		" 9 "	7.1		"
3.2 CR +10LF		" 10 "	7.2		"
3.3 CR +11LF		" 11 "	7.3		"
3.4 CR +12LF		" 12 "	7.4		"
3.5 CR +13LF		" 13 "	7.5		"
3.6 CR +14LF		" 14 "	7.6		"
3.7 CR +15LF		" 15 "	7.7		"

APPENDIX 2 - FILING FAULT NUMBERS

Errors in the use of the filing system give rise to system fault number 20 with b91 set to the filing fault number (see 5.12).

100	Attempt to increase the amount of disc space used above the disc limit.
102	The file owner's directory is full.
103	Archive space allocation is exhausted; the file has been re-classified as P.
104	Attempt at some activity with inadequate status.
106	Attempt to reference a file with an unknown title.
108	Attempt to reference an unknown file owner or the password of an unknown user.
110	Format error in a file title.
112	Using a 1026 extracode with an unassigned value of N.
114	Incorrect attempt to quote a key or password.
116	Attempt to use a 1026 extracode with erroneous parameters.
118	Attempt to use a 1026 extracode with a privileged value of N.
120	Attempt to create or update a file when the owner's disc credit is negative.
122	Attempt to create a relationship with illegal conditions.
200	Attempt to use a file which is already open.
202	Attempt to use a file which is not on disc and is being recovered from a system magnetic tape.
204	The filing system is restarting itself following a system failure, and should be available shortly.

APPENDIX 3 - THE 1026 EXTRACODE

The 1026 extracode provides the majority of basic filing system operations. In each case, the (modified) address N specifies the particular operation, while in general Ba gives the address of up to 4 consecutive words of data in store. Full specifications of these operations are given in the document entitled CAMBRIDGE MULTIPLE-ACCESS SYSTEM - COMMAND SPECIFICATIONS.

The following list gives the values of N and the operations that they represent; other values of N are either unassigned or not permitted to the normal user.

0.4	Change the class of a file
1.0	Delete a file
1.4	Change the status of a file
3.0	Read directory entry for a file
4.0	Read a word of a directory
6.0	Force recovery of a file from a system tape
6.4	Make a file the preferred version
7.0	Check password
7.4	Change password
10.0	Create a relationship
10.4	Remove a relationship
11.0	Quote a key
13.0	Record a protest
13.4	Check a file title
14.0	Request archiving
14.4	Read archive status

APPENDIX 4 - FILE DIRECTORY DETAILS

The following tables give some information about the data held in a file directory. Reliable information concerning parts of the directory not specified here is unavailable.

A file directory consists of a heading followed by a sequence of entries. The heading gives information concerning the directory as a whole, while each entry is one of three types, indicated by the value of the first half-word:

0	unused
0.1	describes a relationship
⩾ 1	describes a file

(other values are used for purposes not described here).

The heading and each entry contain 6 words, the contents of which are described in the following tables. Descriptions are given in terms of 6-bit fields, the position being the word and character address of the first field, and the length being the number of fields. A date and/or time recorded in the tables are each encoded as 2 6-bit fields giving the month and day, or the hour and minute, respectively (January = 1).

Heading

Position	Length	Description
0.0	4	Disc space used
0.4	4	Archive space used
1.0	4	Disc credit
1.4	4	Disc limit
2.0	4	Archive space allocated
2.4	4	Daily income

File entry

Position	Length	Description
0.0	8	Second component of file title (left positioned and filled out with 0.5's)
1.0	8	Third component of file title (as above)
2.0	1	Status for owner category (6 coded bits): 2.0 can update and delete implicitly 1.0 can delete explicitly 0.4 can change status 0.2 can read 0.1 can load
2.1	1	Status for partner category (as above)
2.2	1	Status for key category (as above)
2.3	1	Status for public category (as above)
2.6	2	File length in blocks
4.0	1	6 bits (if 2.0 present - file dumped)
4.3	1	6 bits coded as follows: 4.0 file to be dumped 2.0 file to be removed from disc when dumped 1.0 file is class T 0.4 file is class P or A 0.1 file is preferred version
5.0	4	Date and time of last create or update
5.4	2	Date of last use

Relationship entry

Position	Length	Description
0.0	4	= 0.1
0.4	4	Serial number of relation (if negative, then remaining fields are unspecified)
1.0	1	Status for owner category (6 coded bits): 4.0 can create new files 2.0 can update or delete implicitly 1.0 can delete explicitly 0.4 can change status 0.2 can read 0.1 can load
1.1	1	Status for partner category (as above)
1.2	1	Status for key category (as above)
1.3	1	Status for public category (as above)
1.4	1	Test for parameter in 2.0 3.0 command name = P 2.0 file group = P 1.0 user identifier = P 0.4 key = P 0.2 on line 0.1 password checked
1.5	1	Test for parameter in 3.0 (as above)
1.6	1	Test (no parameter) 0.2 on line 0.1 password checked
1.7	1	Test (no parameter) (as above)
2.0	8	Parameter for first test
3.0	8	Parameter for second test
5.0	4	Date and time relationship was created

APPENDIX 5 - THE MANAGEMENT ALGORITHM

Whenever a user attempts to log in, the management system will determine how this can be done (if at all) having regard to:

(a) The project's allocations and controls.

(b) The size and nature of the current load on the system.

Given that a user's requirements have been checked and found to be in order as far as (a) is concerned, the system determines whether:

(1) the user can be logged in without any consequential action being necessary, or

(2) the user can be logged in, but another user must be given a priority warning, or

(3) the user cannot be logged in.

The following gives details of the algorithm used to determine which of these actions should be taken, and, if another user is to receive a priority warning, which user this should be.

Each project is assigned to an administrative division. For the Titan installation there are 4 divisions as follows:

division 0 Mathematical Laboratory

division 1 Physical Sciences

division 2 Engineering Sciences

division 3 All others

In the following descriptions, access modes and priority levels are denoted by integers m and p respectively, thus:

m	MODE	p	PRIORITY
0	SPECIAL	0	LOW
1	EDIT	1	MEDIUM
2	NORMAL	2	HIGH
3	EXPENSIVE	3	TOP

The following data are preset by the administrative staff:

$U_{m,s}$ = maximum number of users permitted in modes m, m+1, ... and in operating style s.

ST_s = maximum total of store settings for all EXPENSIVE mode users in operating style s.

L_d = load level for division d.

O_d = overload level for division d.

The values of $U_{m,s}$ and ST_s determine whether a new user can log in in a particular mode, without causing another user to be given a priority warning. In assessing the current load to compare with these values, users who have previously been given priority warnings are not counted. If log in is not possible without warning another user off, then the following procedure is followed in an attempt to find a suitable user (if any) who can be removed.

(1) If there are 4 or more users who have been given a priority warning and who have not yet logged out, then the system is considered to be <u>overfull</u>; log in is not permitted.

(2) If there are any users in OVERTIME status (and who have not already received a priority warning), then the system chooses and warns the user that has used the most EXEC since he logged in; log in of the new user is then permitted.

(3) Let N_d be the number of users logged in (and who have not received a priority warning) from division d. Then for each division d, the division is said to be:

underloaded if $N_d < L_d$

loaded if $L_d \leqslant N_d < O_d$

overloaded if $O_d \leqslant N_d$

The divisions are then ranked into an order of (decreasing) precedence, as follows:

(a) The new user's own division if it is overloaded.

(b) Any other overloaded divisions in decreasing order of $(N_d - O_d)$.

(c) The new user's own division if it is not overloaded.

(d) Any other loaded divisions in random order.

(e) Any other underloaded divisions in random order.

The system now searches the users of these divisions in this order until it finds a user with less priority than that of the user trying to log in, and in a mode which enables the correct balance to be maintained according to $U_{m,s}$ and ST_s. The extent of the search undertaken depends on the priority of the new user, as follows:

LOW priority	no search.
MEDIUM priority	through divisions of type (a), (b), (c) only.
HIGH priority	through divisions of type (a), (b), (c), (d) only.
TOP priority	through all divisions.

If, in deciding which user to choose from a particular division, there are

more than 2 to be considered, the system chooses the user who has had the most EXEC since he logged in. If this procedure is successful in finding a suitable user to be warned, log in is permitted, otherwise it is not.

APPENDIX 6 - INDEX OF COMMANDS

The following table gives the names of the most important system commands. Summary descriptions of these commands are given in 7.2, and full user descriptions are given in the separate document entitled CAMBRIDGE MULTIPLE-ACCESS SYSTEM - COMMAND SPECIFICATIONS. A letter 'X' against the name of a command indicates that it is always obeyed interactively, irrespective of the user's access mode. A letter 'E' indicates that the command is available in EDIT mode.

ARCHIVE		MESSAGE	E
AUTOCODE		MLS	
CLOSE	E	PASSWORD	XE
COMMAND	E	PATCH	X
COPY		PDP	
CURRENT	E	PREFER	E
DELETE	E	PRINT	E
EDIT	XE	PROJECT	
ETAL		RECOVER	
EXAMINE	XE	RELATION	XE
E3		RUNJOB	E
FIGARO	X	SAVE	E
FILE	E	SET	E
FINISH	E	SETDUMP	
F4		SIGNAL	E
HELP	E	STATUS	
IIT		STREAMS	
INPUT	XE	TIME	
LISP		TYPE	E
LIST	E	UNLOCK	XE

INDEX

A

@ characters, 7
Access mode, 15,20
Access status, 45,52,55
Allocation, time, 14
Allocation, file space, 49
Archive file, 49,58
Archive space, 51
Archive name, 59
Argument string, 32,37
Auxiliary system command, 69

B

Backing-store device, 3,35,47
Block file, 44
BREAK key, 9
BSD, see Backing-store device

C

Character file, 44
Class of file, 49
COMMAND command, 41
Command name, 32
Command status, 31
Command summaries, 69-79
COMP time, 14,18,21
Console number, 5,12
Controls, 14
Credit bound, 51
CTRL key, 6
Current document, 31

D

Daily income, 50
Directory, 60,84
Disc credit, 50
Disc limit, 51
Division, 15,87
Document, 2
Dump mechanism, 40

E

EDIT mode, 16
Editor, 71
Environment declaration, 32,33
Escaped characters, 7
EXEC time, 14,18,21
EXPENSIVE mode, 16,38
Extracode, 3
Extracode, filing, 83

F

File activity, 52
File class, 49
File dumping, 59
File errors, 61,82
File interlocks, 58
File owner, 13,15
File recovery, 59
File status, 45

I

Input stream, 34,47
Interactive command, 15,31
Internal code, 6,78
Interrupt, 8

J

Job description, 18

K

Key category, 52
Key holder, 55

L

LIMOUT, 20
LIMSTORE, 20
Log-in parameter, 20
Log-in procedure, 21

L contd.

Log out, 27
Log out, automatic, 29

M

Message, 11,27,75
Mode, see Access mode
Modem, 5
Monitor, 8
Multiplexor, 5

N

Non-interactive command, 15,31
NORMAL mode, 16

O

Off-line jobs, 17,43
Operating style, 16
OPTIONS, 22
OVERTIME, 21,27
Output stream, 35,47
Owner category, 52

P

Partner, 54
Partner category, 52
Password, 13,22,51,75
PDP7, PDP9, 12
Phases, 3
Preferred versions, 48
Priority, 20
Private command, 64
Project number, 13,14,22
Public category, 52

Q

Quit, 8

R

Relationship, 53
Remote job entry, 15,17
Restart, 11
Results document, 32
RETURN key, 6

S

Security, 13,51
Shift, 17
SHIFT key, 6
SPECIAL mode, 16
Stream, 2
Style, see Operating style
System command, 69

U

User identifier, 13,21
User status, 56

W

Warnings, 10,28

Printed by E. & E. Plumridge, Linton, Cambridge, England